PassQueen
a.k.a dumpsqueen.com

Developing Solutions for Microsoft Azure

Microsoft AZ-204

Version Demo

Total Demo Questions: 55

Total Premium Questions: 552

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Develop Azure compute solutions	120
Topic 2, Develop for Azure storage	110
Topic 3, Implement Azure security	98
Topic 4, Monitor, troubleshoot, and optimize Azure solutions	53
Topic 5, Connect to and consume Azure services and third-party services	123
Topic 6, Case Study 1	2
Topic 7, Case Study 2	2
Topic 8, Case Study 3	2
Topic 9, Case Study 4	3
Topic 10, Case Study 5	2
Topic 11, Case Study 6	2
Topic 12, Case Study 7	5
Topic 13, Case Study 8	2
Topic 14, Case Study 9	6
Topic 15, Case Study 10	2
Topic 16, Case Study 11	2
Topic 17, Case Study 12	2
Topic 18, Case Study 13	4
Topic 19, Case Study 14	3
Topic 20, Case Study 15	2
Topic 21, Case Study 16	2
Topic 22, Case Study 17	2
Topic 23, Case Study 18	3
Total	552

QUESTION NO: 1

You need to ensure the security policies are met.

What code do you add at line CS07 of ConfigureSSE.ps1?

A. â€“PermissionsToKeys create, encrypt, decrypt

B. â€“PermissionsToCertificates create, encrypt, decrypt

C. â€“PermissionsToCertificates wrapkey, unwrapkey, get

D. â€“PermissionsToKeys wrapkey, unwrapkey, get

ANSWER: D

Explanation:

`â€"PermissionsToKeys wrapkey, unwrapkey, get` grants the minimum Azure Key Vault key permissions needed for a service that uses customer-managed encryption keys. The `get` permission allows the service to retrieve the key's public properties and key material metadata needed to identify and use the configured key. The `wrapKey` permission permits encryption of a data-encryption key with the Key Vault key, while `unwrapKey` permits recovery of that data-encryption key when encrypted data must be read. These are key-operation permissions, so they must be assigned with `-PermissionsToKeys` rather than certificate permissions. Limiting the policy to these operations follows least privilege: the identity receives only the cryptographic operations required to protect and access data, without administrative permissions for creating, importing, deleting, or otherwise managing keys. Azure Key Vault access policies define key permissions independently from secret and certificate permissions, and the supported key permissions include `get`, `wrapKey`, and `unwrapKey`. See Set-AzKeyVaultAccessPolicy documentation and Azure Key Vault security features.

QUESTION NO: 2

You need to secure the Azure Functions to meet the security requirements.

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

NOTE: Each correct selection is worth one point.

A. Store the RSA-HSM key in Azure Cosmos DB. Apery the built-in policies for customer-managed keys and allowed locations.

B. Create a free tier Azure App Configuration instance with a new Azure AD service principal.

C. Store the RSA-HSM key in Azure Key Vault with soft-delete and purge-protection features enabled.

D. Store the RSA-HSM key in Azure Blob storage with an Immutability policy applied to the container.

E. Create a Premium tier Azure App Configuration instance with an assigned Azure AD managed identity.

ANSWER: C E

Explanation:

Store the RSA-HSM key in Azure Key Vault with soft-delete and purge-protection features enabled. Azure Key Vault is the appropriate service for safeguarding an HSM-protected customer-managed encryption key. Soft delete retains deleted vaults, keys, and secrets for the retention period, while purge protection prevents permanent removal during that period. Together, these protections ensure that the encryption key cannot be accidentally or maliciously destroyed, which is essential because loss of a customer-managed key can make encrypted configuration data inaccessible.

Create a Premium tier Azure App Configuration instance with an assigned Azure AD managed identity. Azure App Configuration centralizes application configuration across environments and geographic deployments, and its customer-managed key capability uses a key held in Azure Key Vault. The managed identity gives the App Configuration resource an Azure AD identity through which it can be granted the required access to the Key Vault key, without putting credentials in

application settings or source code. Customer-managed key encryption is supported by the Premium tier of Azure App Configuration. For implementation guidance, see Customer-managed keys for Azure App Configuration and Azure Key Vault soft-delete and purge protection.

QUESTION NO: 3

You are developing an ASP.NET Core Web API web service. The web service uses Azure Application Insights for all telemetry and dependency tracking. The web service reads and writes data to a database other than Microsoft SQL Server.

You need to ensure that dependency tracking works for calls to the third-party database.

Which two Dependency Telemetry properties should you store in the database? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Telemetry.Context.Operation.Id

B. Tetemetry.Context.Cloud.RoleInstance

C. Telemetry.Id

D. Telemetry.ContextSession.Id

E. Telemetry.Name

ANSWER: A C

Explanation:

`Telemetry.Context.Operation.Id` and `Telemetry.Id` provide the correlation information needed when an operation crosses a persistence boundary, such as a third-party database. Application Insights uses the operation ID as the root identifier for a distributed transaction. Persisting `Telemetry.Context.Operation.Id` allows a later process that handles the stored database work to associate its telemetry with the same end-to-end operation or trace. Persisting `Telemetry.Id` preserves the identifier of the specific dependency telemetry item that initiated the stored work. A subsequent operation can use that value as its parent ID, producing a parent-child relationship between the dependency call and telemetry created when the database work is later processed. Together, these values enable Application Insights to reconstruct the causal relationship and display related requests, dependencies, traces, and exceptions as one distributed transaction. This pattern is especially important where automatic dependency instrumentation does not natively support the database provider and custom telemetry or manual context propagation is required. Microsoft documents operation IDs and parent IDs as the core fields used for telemetry correlation and recommends propagating them across asynchronous boundaries. See Distributed tracing telemetry data model and Track custom operations with Application Insights.

QUESTION NO: 4

You need to investigate the http server log output to resolve the issue with the ContentUploadService.

Which command should you use first?

A. az webapp log tail

B. az ams live-output

C. az monitor activity-log

D. az container attach

ANSWER: A

Explanation:

`az webapp log tail` is the appropriate command to begin investigating the HTTP server log output for an Azure App Service application. It streams the application's enabled log output in real time, allowing an engineer to reproduce or observe requests that result in intermittent HTTP 502 responses and correlate them with web-server or application-level events. This provides directly relevant diagnostic evidence from the affected app, rather than only control-plane events.

Before streaming is useful, App Service logging must be enabled and configured to retain the required application and web-server logs. For Linux App Service apps, log streaming is commonly used to view container and application output; for Windows apps, the available output depends on the configured logging providers. The command can be run with the target app name and resource group, for example: `az webapp log tail --name <app-name> --resource-group <resource-group>`. Review the streamed entries around the failing requests, including timestamps, request paths, upstream connection messages, and application exceptions. Microsoft documents log streaming with the Azure CLI in Enable diagnostics logging for apps in Azure App Service and lists the command syntax in the Azure CLI reference.

QUESTION NO: 5

Your company has an Azure Kubernetes Service (AKS) cluster that you manage from an Azure AD-joined device. The cluster is located in a resource group.

Developers have created an application named MyApp. MyApp was packaged into a container image.

You need to deploy the YAML manifest file for the application.

Solution: You install the Azure CLI on the device and run the kubectl apply –f myapp.yaml command.

Does this meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because installing the Azure CLI alone does not establish the prerequisites required to deploy a manifest with `kubectl`. The device must have the Kubernetes command-line client installed and must have a kubeconfig context containing credentials and cluster connection details for the target AKS cluster. For AKS, an authorized user typically runs `az aks get-credentials --resource-group <resource-group> --name <cluster-name>`. This retrieves the cluster access configuration and merges it into the local kubeconfig file. With Azure AD-integrated AKS, the user must also authenticate and be authorized for Kubernetes access, generally through the appropriate Azure role assignments or Kubernetes RBAC configuration. After the kubectl client is available and valid AKS credentials have been acquired, `kubectl apply -f myapp.yaml` can create or update the Kubernetes resources described by the manifest. Azure documents the credential retrieval process in Connect to an AKS cluster and documents installation of the Kubernetes CLI in Install kubectl.

QUESTION NO: 6 - (DRAG DROP)

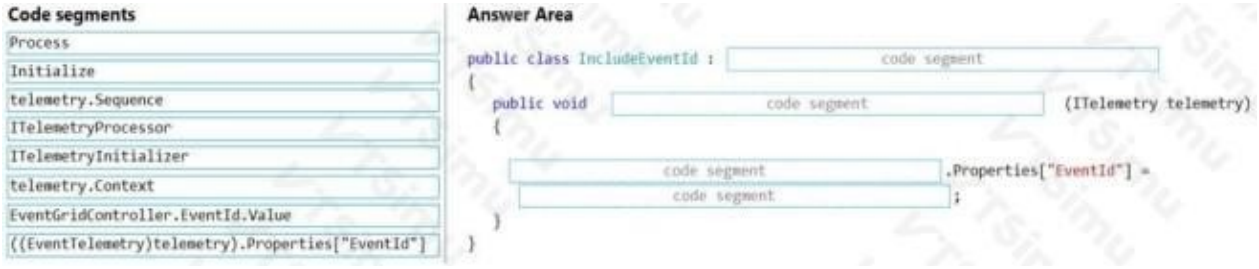
DRAG DROP

You need to ensure that PolicyLib requirements are met.

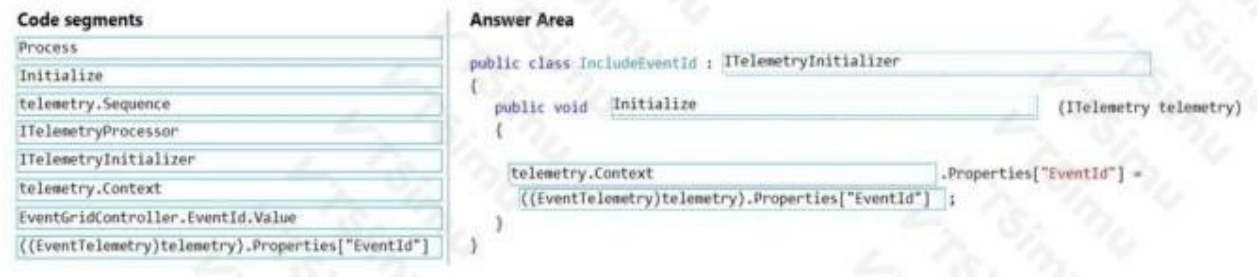
How should you complete the code segment? To answer, drag the appropriate code segments to the correct locations. Each code segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Select and Place:



ANSWER:



Explanation:

The correct implementation creates an Application Insights telemetry initializer by having `IncludeEventId` implement `ITelemetryInitializer`. An initializer is invoked for telemetry items as they are prepared for submission, making it the appropriate shared-library mechanism for enriching telemetry consistently across ASP.NET Core applications and services. The interface requires the `Initialize(ITelemetry telemetry)` method, which receives the telemetry item that is currently being processed.

Within that method, `telemetry.Context` is the common context attached to the outgoing telemetry item. Assigning a value to `telemetry.Context.Properties["EventId"]` adds the event identifier to the telemetry context so that the identifier is available as a custom property when the item reaches Application Insights. This allows telemetry associated with a user-facing event to be correlated and queried using the same event identifier.

The source value is obtained with `((EventTelemetry)telemetry).Properties["EventId"]`. The cast accesses the custom-properties collection on the event telemetry item, where the original EventId was stored. Copying that value into the telemetry context ensures the EventId is carried in the standard context property bag for the telemetry being initialized. This is a suitable pattern for a reusable library because the enrichment logic is centralized rather than repeated throughout every web service.

Microsoft documents telemetry initializers as the supported extensibility point for adding or modifying properties on telemetry before it is sent. See Application Insights telemetry processors and initializers and ITelemetryInitializer interface documentation for the initializer contract and telemetry-enrichment behavior.

QUESTION NO: 7 - (HOTSPOT)

HOTSPOT

You are developing an ASP.NET Core time sheet application that runs as an Azure Web App. Users of the application enter their time sheet information on the first day of every month.

The application uses a third-party web service to validate data.

The application encounters periodic server errors due to errors that result from calling a third-party web server. Each request to the third-party server has the same chance of failure.

You need to configure an Azure Monitor alert to detect server errors unrelated to the third-party service. You must minimize false-positive alerts.

How should you complete the Azure Resource Manager template? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Hot Area:

ANSWER:

Explanation:

Use **DynamicThresholdCriterion**, the **Http5xx** metric, and **Low** alert sensitivity. The application receives a predictable surge of use when time sheets are entered at the start of each month. At the same time, each outbound validation call to the external service has a recurring probability of producing a server-side failure. Consequently, the number of server-error responses that the web app returns is not expected to remain at one fixed number: it naturally changes with the request volume and with the normal rate of failures from the dependency.

A dynamic threshold criterion is designed for this situation because Azure Monitor uses prior metric behavior to establish the expected range and identifies values that materially deviate from that learned pattern. This allows normal, recurring Http5xx

activity—including periods with increased monthly traffic—to be treated as baseline behavior. An unusually elevated server-error rate can then be detected as an anomaly, which is the useful signal for a problem not explained by the ordinary third-party failure pattern. Azure Monitor documents dynamic thresholds and their behavior at Dynamic thresholds in metric alerts.

The Http5xx metric is the appropriate signal because it measures HTTP responses with server-error status codes from the web application. It therefore directly monitors the class of failures specified in the requirement. App Service metric definitions, including HTTP status-code metrics, are documented at Monitoring data reference for Azure App Service.

Finally, selecting Low sensitivity intentionally requires a more significant departure from the dynamic baseline before an alert is triggered. That wider tolerance is appropriate when avoiding noisy alerts is a priority and when some level of dependency-related failure is already expected. It preserves alerting for substantial, unusual increases while minimizing false-positive notifications.

QUESTION NO: 8

You develop an ASP.NET Core app that uses Azure App Configuration. You also create an App Configuration containing 100 settings. The app must meet the following requirements:

• Ensure the consistency of all configuration data when changes to individual settings occur.

• Handle configuration data changes dynamically without causing the application to restart.

• Reduce the overall number of requests made to App Configuration APIs.

You must implement dynamic configuration updates in the app.

What are two ways to achieve this goal? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Increase the App Configuration cache expiration from the default value.

B. Create and implement environment variables for each App Configuration store setting.

C. Decrease the App Configuration cache expiration from the default value.

D. Register all keys in the App Configuration store. Set the refreshAll parameter of the Register method to false.

E. Create and register a sentinel key in the App Configuration store. Set the refreshAll parameter of the Register method to true.

F. Create and configure Azure Key Vault. Implement the Azure Key Vault configuration provider.

ANSWER: A E

Explanation:

Increasing the App Configuration cache expiration from the default value reduces the frequency with which the provider checks App Configuration for updates. Dynamic refresh does not query the service on every request; instead, refresh activity is governed by the configured refresh interval. A longer interval therefore lowers the overall number of App Configuration API requests while retaining the ability to apply discovered changes without restarting the ASP.NET Core application.

Creating and registering a sentinel key in the App Configuration store, with the `refreshAll` parameter set to `true`, provides a consistent way to publish related configuration changes. Update the individual settings first, then update the sentinel key only after all related changes are complete. When the provider detects that the sentinel has changed, `refreshAll: true` causes it to reload the selected configuration values together. This avoids an application observing a partially updated set of dependent settings and enables runtime configuration updates without an application restart. Microsoft documents the sentinel pattern specifically for ensuring consistency across multiple configuration values and documents refresh intervals as the mechanism for controlling refresh-check frequency. See Dynamic configuration in ASP.NET Core and Azure App Configuration refresh best practices.

QUESTION NO: 9

You need to test the availability of the corporate website.

Which two test types can you use?

A. Custom testing using the TrackAvailability API method

B. Standard

C. URL Ping

D. Multi-step

ANSWER: A B

Explanation:

Custom testing using the TrackAvailability API method and Standard are supported approaches for monitoring a corporate website's availability with Application Insights. A standard availability test is a managed, single-request web test that runs from Azure's test locations at a recurring interval. It checks whether the target endpoint responds successfully and records availability results, response time, and diagnostic details that can be used for alerting and investigation. This makes it appropriate for routinely validating that a public website or endpoint can be reached by users.

Custom testing using the TrackAvailability API method is appropriate when the application needs to define and report its own availability measurement. Code can perform a tailored check—such as validating a dependency, executing a specialized workflow, or testing a resource that cannot be evaluated through the managed web-test model—and then send an availability result to Application Insights through telemetry. The reported measurements are retained as availability telemetry and can be queried, visualized, and used in alerts alongside managed test results. Microsoft documents standard tests as the current managed availability-test experience and documents `TrackAvailability` for submitting custom availability results. See Standard availability tests and TrackAvailability telemetry.

QUESTION NO: 10

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

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

You are developing an Azure solution to collect point-of-sale (POS) device data from 2,000 stores located throughout the world. A single device can produce 2 megabytes (MB) of data every 24 hours. Each store location has one to five devices that send data.

You must store the device data in Azure Blob storage. Device data must be correlated based on a device identifier. Additional stores are expected to open in the future.

You need to implement a solution to receive the device data.

Solution: Provision an Azure Notification Hub. Register all devices with the hub.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because Azure Notification Hubs does not provide a device-to-cloud data-ingestion mechanism for receiving and storing POS telemetry or files. Notification Hubs is designed to send highly scalable, cross-platform push notifications *to* mobile devices through platform notification services. Registering devices creates registrations that Notification Hubs can

target when an application needs to deliver a notification; it does not establish an endpoint through which those devices can upload their daily data payloads to Azure Blob Storage.

The stated goal requires an ingestion service that accepts data sent by many distributed devices, supports future growth, and enables the application to associate each incoming payload with its device identifier before storing it in Blob Storage. Azure messaging and IoT ingestion services can be used to receive device-originated messages, after which a consumer can persist the data and its device metadata to Blob Storage. Notification Hubs does not perform this receiving, correlation, or storage workflow. Therefore, provisioning a hub and registering devices alone does not meet the requirement. Microsoft describes Notification Hubs as infrastructure for sending push notifications in its Notification Hubs overview. For guidance on selecting Azure messaging services for event and device-data scenarios, see Choose between Azure messaging services.

QUESTION NO: 11

You need to configure all site configuration settings for the corporate website.

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

NOTE: Each correct selection is worth one point.

A. Create a managed identity.

B. Update the role assignments for the Azure App Configuration store

C. Create an Azure Key Vault.

D. Create an Azure App Configuration store.

E. Update the role assignments for the Azure Key Vault.

ANSWER: A B D

Explanation:

Creating a managed identity, creating an Azure App Configuration store, and updating the role assignments for the Azure App Configuration store are the required actions to make the website consume its centrally managed configuration settings securely. Azure App Configuration provides a dedicated configuration store for application settings, including key-value settings and feature flags. After the store is created, the website needs an identity with which to authenticate to Azure rather than embedding a connection string or credential in code or deployment settings. A managed identity supplies that identity and is managed by Azure. The managed identity must then be authorized on the App Configuration store by assigning an appropriate data-plane role, typically *App Configuration Data Reader*, so that the running website can read configuration values. These steps establish the configuration store, the application identity, and the authorization path required at runtime. Microsoft recommends Microsoft Entra ID authentication with managed identities and Azure role-based access control when applications access App Configuration. See Azure App Configuration overview and Azure App Configuration data access authorization.

QUESTION NO: 12

You develop an order-processing application that sends messages to an Azure Service Bus queue.

Network retries can cause the application to send the same order message more than once. The queue must discard duplicate messages that have the same order identifier within a 10-minute interval.

You need to configure the solution.

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

NOTE: Each correct selection is worth one point.

A. Enable duplicate detection and configure a 10-minute duplicate detection history window.

B. Set the MessageId property to the order identifier.

C. Enable sessions on the queue.

D. Set the DeliveryCount property to the order identifier.

E. Create a correlation filter for the queue.

ANSWER: A B

Explanation:

Enable **duplicate detection** for the Service Bus queue and configure a 10-minute duplicate detection history window. Service Bus retains message identifiers that it has accepted during the configured window and discards subsequently received messages with the same identifier.

Set each message's **MessageId** to the order identifier. Duplicate detection evaluates the MessageId value, so the producer must assign a stable identifier that is the same for all retries of the same logical order message.

Duplicate detection protects against duplicate sends within its configured window. Consumer code should still be designed to be idempotent because duplicate processing can occur in other failure scenarios. See Duplicate detection in Azure Service Bus.

QUESTION NO: 13

You need to secure the Azure Functions to meet the security requirements.

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

NOTE: Each correct selection is worth one point.

A. Store the RSA-HSM key in Azure Key Vault with soft-delete and purge-protection features enabled.

B. Store the RSA-HSM key in Azure Blob storage with an immutability policy applied to the container.

C. Create a free tier Azure App Configuration instance with a new Azure AD service principal.

D. Create a standard tier Azure App Configuration instance with an assigned Azure AD managed identity.

E. Store the RSA-HSM key in Azure Cosmos DB. Apply the built-in policies for customer-managed keys and allowed locations.

ANSWER: A D

Explanation:

Store the RSA-HSM key in Azure Key Vault with soft-delete and purge-protection features enabled. Azure Key Vault is the appropriate service for retaining and protecting cryptographic keys, including HSM-protected RSA keys. Soft delete preserves deleted vault objects for a retention period, and purge protection prevents permanent removal during that period. Together, these controls provide recoverability and protect the key material that is required to decrypt the centrally managed configuration data.

Create a standard tier Azure App Configuration instance with an assigned Azure AD managed identity. Azure App Configuration centralizes application settings and feature configuration across environments and regions. Its Standard tier supports customer-managed encryption keys. The managed identity gives the App Configuration resource an Azure AD identity that can be granted access to the Key Vault key without placing credentials or key-access secrets in the Azure Functions application configuration. After granting that identity the required Key Vault key permissions, App Configuration can use the organization-controlled RSA key for encryption at rest.

See Customer-managed keys for Azure App Configuration and Azure Key Vault soft-delete and purge protection.

QUESTION NO: 14

You develop and deploy an Azure App Service web app. The app is deployed to multiple regions and uses Azure Traffic Manager. Application Insights is enabled for the app.

You need to analyse app uptime for each month.

Which two solutions will achieve the goal? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Azure Monitor logs

B. Application Insights alerts

C. Azure Monitor metrics

D. Application Insights web tests

ANSWER: A D

Explanation:

Application Insights web tests provide the synthetic availability monitoring needed to measure whether the application can be reached and responds successfully over time. Tests run from external locations at configured intervals and create an availability result for every execution. For a web app deployed in multiple regions behind Azure Traffic Manager, a URL ping test or a standard test can validate the public endpoint that users access, thereby measuring end-to-end availability rather than only the health of an individual deployment.

Azure Monitor logs provide the historical analysis capability. Application Insights stores web-test execution data in the `availabilityResults` table. A Log Analytics query can group these results by calendar month and calculate the percentage of successful test executions, which represents uptime for each month. For example, queries can summarize successful and total test runs by month and test location, allowing reporting across the required period and investigation of regional behavior when needed. Log retention should be configured to cover the reporting interval.

Microsoft documents availability tests and their resulting telemetry in Availability tests in Application Insights. The schema and query guidance for the underlying availability telemetry are available in the AvailabilityResults table reference.

QUESTION NO: 15

A company maintains multiple web and mobile applications. Each application uses custom in-house identity providers as well as social identity providers.

You need to implement single sign-on (SSO) for all the applications.

What should you do?

A. Use Azure Active Directory B2C (Azure AD B2C) with custom policies. Most Voted

B. Use Azure Active Directory B2B (Azure AD B2B) and enable external collaboration.

C. Use Azure Active Directory B2C (Azure AD B2C) with user flows.

D. Use Azure Active Directory B2B (Azure AD B2B).

ANSWER: A

Explanation:

Use Azure Active Directory B2C (Azure AD B2C) with custom policies. Azure AD B2C is designed for customer-facing web and mobile applications that need to authenticate users through local accounts, social identity providers, and federated external identity providers. Custom policies provide the required flexibility to integrate custom in-house identity providers by defining tailored authentication journeys and claims transformations.

For SSO, Azure AD B2C maintains a session after a user successfully authenticates. Applications that use the same Azure AD B2C tenant and compatible policy configuration can use that session to avoid prompting the user to authenticate again. Custom policies provide explicit controls for session behavior, including the Azure AD B2C session provider, session persistence, and scope. This supports a consistent sign-in experience across the company's web and mobile applications while allowing each application to rely on the same centralized customer identity platform.

Microsoft documents Azure AD B2C session management and SSO configuration for custom policies in Single sign-on session management in Azure AD B2C custom policies. For federation with external identity providers, see Define an OpenID Connect technical profile in a custom policy.

QUESTION NO: 16

You are updating an application that stores data on Azure and uses Azure Cosmos DB for storage. The application stores data in multiple documents associated with a single username.

The application requires the ability to update multiple documents for a username in a single ACID operation.

You need to configure Azure Cosmos DB.

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

NOTE: Each correct selection is worth one point.

A. Configure Azure Cosmos DB to use the Azure Cosmos DB for Apache Gremlin API.

B. Configure Azure Cosmos DB to use the Azure Cosmos DB for MongoDB API.

C. Create a collection sharded on username to store documents.

D. Create an unsharded collection to store documents.

ANSWER: B D

Explanation:

Configure Azure Cosmos DB to use the Azure Cosmos DB for MongoDB API and create an unsharded collection to store documents. The Azure Cosmos DB for MongoDB API provides MongoDB-compatible functionality for multi-document transactions. A transaction groups multiple read and write operations so that they are committed as one atomic unit, providing the required ACID behavior: either all updates for the username are committed or none are committed if the transaction fails.

For the transaction capability addressed by this configuration, the documents must be placed in an unsharded collection. This allows the related document updates to execute within the supported transaction scope, rather than requiring coordination across shards. The application can then begin a session, start a transaction, perform each document update associated with the username, and commit the transaction as a single operation. This directly meets the requirement to update several documents together while maintaining transactional consistency.

Microsoft documents transaction support and its scope for the Azure Cosmos DB for MongoDB API at Use multi-document transactions in Azure Cosmos DB for MongoDB. General API capabilities and compatibility information are available in the Azure Cosmos DB for MongoDB documentation.

QUESTION NO: 17

You ate developing a web application that uses the Microsoft identity platform to authenticate users and resources. The web application calls several REST APIs.

The APIs require an access token from the Microsoft identity platform.

You need to request a token.

Which three properties should you use? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Application secret

B. Redirect URI/URL

C. Application name

D. Supported account type

E. Application ID

ANSWER: A B E

Explanation:

For a server-side web application that acquires access tokens through the Microsoft identity platform, the application registration must provide the client identity and, as a confidential client, authenticate itself when redeeming an authorization code. **Application ID**, also called the client ID, identifies the registered application and is sent as the `client_id` parameter in token requests. **Application secret** is a client credential that proves the web application's identity to the token endpoint; it must be protected and used only on the server, never exposed to a browser. A certificate can also be used as a client credential, but the application secret listed here is valid.

Redirect URI/URL is required for the authorization-code flow used by web apps that sign in users and call APIs on their behalf. Microsoft Entra ID returns the authorization response to this registered URI, and the redirect URI supplied during code redemption must match one configured in the app registration. Together, these settings enable the app to initiate authorization, receive an authorization code, and exchange that code for an access token for the requested REST API scopes. See Microsoft identity platform authorization code flow and Register an application with the Microsoft identity platform.

QUESTION NO: 18

You develop a web application that uses Azure Blob storage for document downloads.

Documents must be accessible only to users who have authenticated by using Microsoft Entra ID. The application must not use storage account access keys or shared access signatures.

You need to grant users read-only access to the documents.

What should you do?

A. Assign the Storage Blob Data Reader role to the users.

B. Assign the Reader role to the users.

C. Generate a service shared access signature with read permission.

D. Provide the storage account primary access key to the users.

ANSWER: A

Explanation:

Assign the users the **Storage Blob Data Reader** Azure RBAC role at the required storage-account, container, or blob scope. This role authorizes Microsoft Entra ID identities to read blob data. Clients can authenticate by using Microsoft Entra ID and obtain an OAuth 2.0 access token for Azure Storage without requiring an account key or SAS token.

The Reader role provides management-plane access to view resource configuration but does not grant permission to read blob data. Storage Blob Data Contributor would grant unnecessary write permissions. For more information, see Assign an Azure role for access to blob data.

QUESTION NO: 19

You are developing a web application that uses the Microsoft identity platform to authenticate users and resources, The web application calls several REST APIs.

The APIs require an access token from the Microsoft identity platform.

You need to request a token.

Which three properties should you use? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Application name

B. Application secret

C. Application ID

D. Supported account type

E. Redirect URI/URL

ANSWER: B C E

Explanation:

A web application that obtains an access token on behalf of a signed-in user is a confidential client and commonly uses the OAuth 2.0 authorization-code flow with the Microsoft identity platform. The **Application ID**, also called the client ID, identifies the app registration in token requests. The **Application secret**, also called a client secret, authenticates the confidential web application when it redeems an authorization code at the token endpoint; it must be protected and never exposed to browsers or source control. The **Redirect URI/URL** identifies the registered callback location to which Microsoft Entra ID returns the authorization response after the user signs in and grants consent. The value sent in the authorization-code flow must match a redirect URI registered for the application. Together, these properties establish the application identity, authenticate the server-side client, and provide the approved callback endpoint needed to obtain tokens for the REST API scopes. Microsoft documents the authorization-code flow and its token request parameters at Microsoft identity platform authorization code flow. Guidance for registering web applications, including redirect URIs and credentials, is available at Quickstart: Register an application with the Microsoft identity platform.

QUESTION NO: 20

You need to support the requirements for the Shipping Logic App.

What should you use?

A. Azure Active Directory Application Proxy

B. Point-to-Site (P2S) VPN connection

C. Site-to-Site (S2S) VPN connection

D. On-premises Data Gateway

ANSWER: D

Explanation:

On-premises Data Gateway is the appropriate choice because it provides the supported bridge between a cloud-hosted Azure Logic App and data sources or systems that remain inside the corporate network. The gateway is installed on a machine within the on-premises environment and establishes outbound, encrypted connections to Azure, so the Logic App can access supported on-premises resources without exposing those resources directly to the public internet. This is particularly important while legacy applications are still in use and during a staged migration from BizTalk-based integrations. The gateway can be used by Logic Apps connectors that support on-premises connectivity, allowing workflows to continue

interacting with internal systems while business processes are modernized. It complements, rather than replaces, the Logic App's enterprise integration capabilities for B2B processing such as X12 messages. Microsoft documents the gateway as the mechanism that enables Logic Apps to access on-premises data sources through supported connectors and describes its secure outbound communication model. See Connect to on-premises data sources from Azure Logic Apps and What is an on-premises data gateway?.

QUESTION NO: 21

You need to monitor ContentUploadService according to the requirements.

Which command should you use?

A. az monitor metrics alert create –n alert –g … - -scopes … - -condition "avg Percentage CPU > 8"

B. az monitor metrics alert create –n alert –g … - -scopes … - -condition "avg Percentage CPU > 800"

C. az monitor metrics alert create --name alert --resource-group … --scopes … --condition "avg CPU Usage > 800"

D. az monitor metrics alert create –n alert –g … - -scopes … - -condition "CPU Usage > 8"

ANSWER: C

Explanation:

The appropriate command uses the `CPU Usage` metric with an average aggregation and a threshold of 800. For Azure Container Instances, CPU usage is reported in millicores. One CPU core equals 1,000 millicores, so 80 percent of one available core equals 800 millicores. Therefore, an alert condition of `avg CPU Usage > 800` raises an alert when the service exceeds the required CPU-consumption threshold. The `az monitor metrics alert create` command creates a metric alert rule for the resource specified through `--scopes`; the resource group and alert name are supplied through `--resource-group` and `--name`. Azure CLI metric-alert conditions use an aggregation, metric name, comparison operator, and numeric threshold, making `avg CPU Usage > 800` the required condition format. Microsoft documents the CPU Usage metric for container groups as a millicore measurement, and documents metric alert creation and condition syntax in the Azure CLI reference. See supported Azure Container Instances metrics and az monitor metrics alert.

QUESTION NO: 22

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

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

You develop Azure solutions.

You must grant a virtual machine (VM) access to specific resource groups in Azure Resource Manager.

You need to obtain an Azure Resource Manager access token.

Solution: Use the Reader role-based access control (RBAC) role to authenticate the VM with Azure Resource Manager.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because assigning the Reader RBAC role does not authenticate a virtual machine or issue an Azure Resource Manager access token. Azure RBAC is an authorization mechanism: after an identity has obtained a token, a role assignment determines which Azure Resource Manager operations that identity can perform at the assigned scope. In this case, Reader can be assigned to the VM's managed identity at the required resource-group scopes to grant read permissions, but the assignment itself is not the token-acquisition process.

To obtain an Azure Resource Manager token from code running on an Azure VM, enable a system-assigned or user-assigned managed identity for the VM and request a token from the Azure Instance Metadata Service (IMDS). The request specifies the Azure Resource Manager audience, `https://management.azure.com/`. IMDS returns an OAuth 2.0 access token for the managed identity, which the application can then include as a bearer token in Resource Manager API requests. The managed identity must separately have an appropriate RBAC role assignment on each required resource group. Microsoft documents token retrieval through IMDS for managed identities and the distinct role-assignment model in Use managed identities on a virtual machine to acquire an access token and What is Azure role-based access control?.

QUESTION NO: 23

You develop and deploy a Java RESTful API to Azure App Service.

You open a browser and navigate to the URL for the API. You receive the following error message:

```
Failed to load http://api.azurewebsites.net:6000/#/api/Products: No 'Access-
Control-Allow-Origin'header is present on the requested resource.
Origin 'http://localhost:6000' is therefore not allowed access
```

You need to resolve the error.

What should you do?

A. Bind an SSL certificate

B. Enable authentication

C. Enable CORS

D. Map a custom domain

E. Add a CDN

ANSWER: C

Explanation:

Enable CORS is the appropriate action because Cross-Origin Resource Sharing controls whether browser-based code hosted at one origin can call an API hosted at a different origin. An origin consists of the scheme, host, and port, so a web application and an Azure App Service API commonly have different origins even when they belong to the same solution. Browsers enforce the same-origin policy and block a cross-origin request unless the API response includes an appropriate `Access-Control-Allow-Origin` header. Azure App Service provides a CORS configuration feature that adds the required response headers for the allowed origins. Configure the specific origin of the client application, such as its HTTPS URL, rather than broadly allowing every origin in a production environment. After the client origin is added to the App Service CORS allowed-origins list, browser requests can proceed when the API otherwise permits them. This configuration is particularly applicable to Java REST APIs because the restriction is imposed by the browser, not by the Java runtime or REST framework. See Azure App Service CORS functionality and App Service security documentation.

QUESTION NO: 24

You are developing an e-commerce solution that uses a microservice architecture.

You need to design a communication backplane for communicating transactional messages between various parts of the solution. Messages must be communicated in first-in-first-out (FIFO) order.

What should you use?

A. Azure Storage Queue

B. Azure Event Hub

C. Azure Service Bus queues with sessions enabled

D. Azure Event Grid

ANSWER: C

Explanation:

Azure Service Bus queues with sessions enabled are appropriate for transactional communication among microservices when ordered processing is required. Azure Service Bus is an enterprise messaging broker designed to decouple services while supporting durable, reliable command and transaction-oriented messaging. A queue provides competing-consumer processing, so a message is handled by one receiver, which suits work items such as order submission, inventory reservation, payment processing, and fulfillment requests.

For a strict first-in-first-out guarantee, the sender assigns related messages a common session ID and the queue has sessions enabled. Service Bus sessions provide ordered handling of messages within a session: a receiver obtains an exclusive lock on that session and processes its messages sequentially. This lets the solution preserve the order for each logical stream, such as all messages associated with a particular order, while still allowing separate sessions to be processed concurrently for throughput.

Service Bus also provides capabilities commonly needed for transactional microservice backplanes, including dead-letter queues, duplicate detection, delivery retries, scheduled delivery, and transactions. These features make it well suited to dependable business-message workflows. See Message sessions in Azure Service Bus and Azure Service Bus queues, topics, and subscriptions.

QUESTION NO: 25

You develop a serverless application that processes images uploaded to Azure Blob storage.

The application must receive a notification when an image is uploaded. If the notification cannot be delivered to the processing endpoint after retries are exhausted, the undelivered event must be retained for later investigation.

You need to configure Azure Event Grid.

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

NOTE: Each correct selection is worth one point.

A. Create an event subscription for the Microsoft.Storage.BlobCreated event type.

B. Configure an Azure Storage blob container as the dead-letter destination.

C. Configure an Azure Storage queue as the dead-letter destination.

D. Enable blob versioning on the storage account.

E. Create an Event Grid custom topic for the storage account.

ANSWER: A B

Explanation:

Create an Event Grid subscription for the **Microsoft.Storage.BlobCreated** event type. Azure Blob Storage publishes this event when a blob is created, and the subscription routes the event to the image-processing endpoint.

Configure a **dead-letter destination** that uses an Azure Storage blob container. Event Grid writes undelivered events to the configured Storage Blob container when it cannot deliver an event within its retry policy. The retained event data can then be examined or reprocessed. A Storage queue can be an event handler, but it is not the Event Grid dead-letter destination type. See Event Grid message delivery and retry and Dead-letter and retry policies for Event Grid.

QUESTION NO: 26

You are developing an Azure Function App that runs in an App Service Plan. The Azure Function is triggered by a Timer object. You observe that the Azure Function does not reliably trigger when scheduled. Which two actions should you perform?

A. Verify that Always On is enabled.

B. Modify the trigger to use a SignaIR trigger.

C. Ensure that the function has a retry configured.

D. Modify the trigger to use Consumption mode instead of the App Service plan.

ANSWER: A C

Explanation:

Verify that Always On is enabled. is required for a timer-triggered function running in an App Service plan. Without Always On, the App Service host can become idle after a period without incoming HTTP traffic. An idle host is not continuously available to monitor and execute timer schedules, so enabling Always On keeps the Functions host loaded and allows scheduled timer invocations to occur consistently. Microsoft specifically identifies Always On as necessary for timer triggers in non-Consumption hosting plans.

Ensure that the function has a retry configured. improves the reliability of the scheduled workload when an invocation starts but encounters a transient failure. Azure Functions supports retry policies for Timer triggers. A configured retry policy causes the runtime to reattempt a failed execution according to the selected fixed-delay or exponential-backoff policy, helping ensure that temporary dependency, network, or service failures do not prevent the scheduled work from completing. The retry policy is configured at the function level and applies when an execution fails with an uncaught exception.

See the Azure Functions Timer trigger documentation and Azure Functions retry guidance.

QUESTION NO: 27 - (HOTSPOT)

You are developing an application that runs in several customer Azure Kubernetes Service clusters, Within each cluster, a pod runs that collects performance data to be analyzed later, a large amount of data is collected so saving latency must be minimized

The performance data must be stored so that pod restarts do not impact the stored data. Write latency should be minimized.

You need to configure blob storage.

How should you complete the YAML configuration? To answer, select the appropriate options in the answer area.

ANSWER:

Explanation:

A `StorageClass` is the Kubernetes resource that defines how persistent volumes are dynamically provisioned for workloads. This is the correct resource type when the application needs storage that survives a pod restart, because a pod can reconnect through a PersistentVolumeClaim to storage provisioned independently of the pod lifecycle. Kubernetes documents that a StorageClass describes the classes of storage offered by a cluster administrator and supports dynamic provisioning through its provisioner field. See the Kubernetes StorageClass documentation.

For this workload, the correct provisioner is `azure-disk`. Azure managed disks provide persistent block storage for AKS pods. Premium_LRS identifies Premium SSD managed disks, which are designed for I/O-intensive workloads and provide the low latency needed when a large amount of performance data is written. Microsoft's AKS documentation describes Azure Disks as high-performance, durable block-level storage and identifies Premium SSD as appropriate for production and performance-sensitive workloads. See Storage options for applications in AKS and Azure managed disk types.

The reclaim policy must be `retain`. This preserves the underlying Azure managed disk when the associated claim is deleted, preventing automatically reclaimed storage from removing collected performance data. Retaining the volume supports the requirement that stored data remain available independently of pod restarts and allows the data to be

recovered or managed deliberately. Kubernetes documents that the Retain reclaim policy keeps the volume and its data after release. See Persistent volume reclaim policies.

QUESTION NO: 28

You deploy an API to API Management

You must secure all operations on the API by using a client certificate.

You need to secure access to the backend service of the API by using client certificates.

Which two security features can you use?

A. Azure AD token

B. Self-signed certificate

C. Certificate Authority (CA) certificate

D. Triple DES (3DES) cipher

E. Subscription key

ANSWER: B C

Explanation:

Azure API Management can authenticate to a backend service by presenting a client certificate during the TLS handshake, enabling mutual TLS authentication. A self-signed certificate can be used when the backend is configured to trust that specific certificate, which is common for controlled internal or development-to-production integrations. A Certificate Authority (CA) certificate can also be used where the client certificate presented by API Management is issued under a trusted CA chain, allowing the backend to validate the identity of the API Management gateway through standard certificate-chain validation.

In API Management, the certificate is uploaded or referenced in the API Management instance and associated with the backend request through the `authentication-certificate` policy. This policy supports selecting a certificate by its thumbprint or certificate identifier and causes API Management to present it to the backend. The backend must be configured to request and validate client certificates, including validation of the trusted issuer or the explicitly trusted self-signed certificate. This provides cryptographic proof of the caller's identity and protects the API Management-to-backend connection with mutual authentication. See API Management authentication-certificate policy and How to secure APIs using client certificate authentication in API Management.

QUESTION NO: 29

You develop and deploy an ASP.NET Core application that connects o an Azure Database for MySQL instance.

Connections to the database appear to drop intermittently and the application code does not handle the connection failure.

You need to handle the transient connection errors in code by implementing retries.

What are three possible ways to achieve this goal? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Increase connection repeat attempts exponentially up to 120 seconds.

B. Close the database connection and immediately report an error.

C. Wait five seconds before repeating the connection attempt to the database.

D. Disable connection pooling and configure a second Azure Database for MySQL instance.

E. Set a maximum number of connection attempts to 10 and report an error on subsequent connections.

ANSWER: A C E

Explanation:

Transient database connectivity failures should be handled with a bounded retry policy rather than treated as immediately fatal. “Increase connection repeat attempts exponentially up to 120 seconds” implements exponential backoff, which progressively increases the delay between attempts and reduces pressure on the service while it recovers. “Wait five seconds before repeating the connection attempt to the database” is also a valid fixed-delay retry approach for intermittent failures. “Set a maximum number of connection attempts to 10 and report an error on subsequent connections” ensures that retries are bounded, so the application does not retry indefinitely and can return a controlled failure after the configured attempt limit is reached. In production, the retry policy should be applied only to errors known to be transient, and the application should log the final failure for investigation. Microsoft’s retry guidance recommends selecting an appropriate retry count and delay strategy, including fixed or exponential delays, based on the operation and expected recovery time. See the Transient fault handling guidance and the Retry pattern.

QUESTION NO: 30 - (SIMULATION)

You need to implement event routing for retail store location data.

Which configuration should you use?

ANSWER: See the explanation for the answer

Explanation:

Use **Azure Event Grid** to route the location events. Create an **Event Grid event subscription** on the source topic/system topic and configure the subscription filter for the retail-store location event type (and, where applicable, a subject prefix identifying the store/location). Set the required processing target, such as an Azure Function or Event Hub, as the subscription endpoint.

This is preferable to having the event producer send directly to each consumer: Event Grid provides the routing layer and supports event-type/subject filtering so that only retail store location events reach the destination.

QUESTION NO: 31

You develop Azure solutions.

You must connect to a No-SQL globally-distributed database by using the .NET API.

You need to create an object to configure and execute requests in the database.

Which code segment should you use?

A. new Container(EndpointUri, PrimaryKey);

B. new Database(Endpoint, PrimaryKey);

C. new CosmosClient(EndpointUri, PrimaryKey);

ANSWER: C

Explanation:

`new CosmosClient(EndpointUri, PrimaryKey);` is correct because `CosmosClient` is the primary entry point for the Azure Cosmos DB .NET SDK. It represents a client-side, thread-safe object used to configure connectivity to an Azure Cosmos DB account and perform operations against databases, containers, and items. The constructor accepts the account endpoint URI and a credential, such as an account key; in this case, `EndpointUri` identifies the Cosmos DB account and

`PrimaryKey` authenticates the request. After creating this client, an application can obtain database and container references and execute asynchronous operations, including creating resources, reading items, querying data, and writing documents.

For production applications, Microsoft recommends creating a single `CosmosClient` instance and reusing it for the lifetime of the application. Reuse allows the SDK to manage connections efficiently and avoid unnecessary connection establishment overhead. The same client can also be configured with `CosmosClientOptions` when requirements include a preferred region, custom serialization, retry behavior, or connection settings. Microsoft documents `CosmosClient` as the client used to interact with the Azure Cosmos DB service and provides examples that initialize it with an endpoint and key. See Get started with Azure Cosmos DB for NoSQL using .NET and the CosmosClient API reference.

QUESTION NO: 32

You are developing an Azure Durable Function to manage an online ordering process.

The process must call an external API to gather product discount information.

You need to implement Azure Durable Function.

Which Azure Durable Function types should you use? Each correct answer presents part of the solution

NOTE: Each correct selection is worth ore point

A. Orchestrator

B. Entity

C. Activity

D. Client

ANSWER: A C

Explanation:

An **Orchestrator** function is appropriate for managing the online ordering workflow. It defines the durable, stateful sequence of work, such as requesting discount information and then continuing with subsequent ordering steps. Durable Functions persists orchestration state and can reliably resume the workflow after restarts, retries, or scale events. The orchestrator should coordinate the workflow rather than directly perform nondeterministic I/O operations.

An **Activity** function should perform the external API call that retrieves product discount information. Activity functions are intended for the individual units of work within an orchestration, including calling external services, performing I/O, or executing business logic. The orchestrator invokes the activity and awaits its durable result, allowing the overall process to remain reliable and replay-safe. This separation is important because orchestrator code must be deterministic, while an activity function is the proper execution boundary for an external HTTP request.

For more information, see Durable Functions types and features overview and Durable Functions orchestrator code constraints.

QUESTION NO: 33

You need to implement a solution to resolve the retail store location data issue.

Which three Azure Blob features should you enable? Each correct answer presents part of the solution.

NOTE Each correct selection is worth one point

A. Immutability

B. Snapshots

C. Versioning

D. Soft delete

E. Object replication

F. Change feed

ANSWER: C D F

Explanation:

Versioning, **Soft delete**, and **Change feed** are the required features for Azure Blob Storage point-in-time restore. Point-in-time restore lets a storage account be restored to a previous state within its configured retention period, which is suitable when accidental deletion or unintended changes affect retail store location data.

Versioning preserves prior versions of a blob when that blob is overwritten or deleted, allowing the restore process to reconstruct the account state at the selected time. **Soft delete** retains deleted blobs, blob versions, and snapshots for a specified retention period instead of permanently removing them immediately. This retention is essential to recovering data that was deleted unexpectedly. **Change feed** provides an ordered, durable log of changes made to blobs. Azure uses this log to identify and replay the changes needed to return containers to their state at the requested restore point.

These features must be enabled before point-in-time restore can be configured. The storage account must also meet the supported account and redundancy requirements described in the Azure Blob Storage documentation. See Point-in-time restore for block blobs and Blob versioning.

QUESTION NO: 34

You need to deploy the CheckUserContent Azure Function. The solution must meet the security and cost requirements.

Which hosting model should you use?

A. Premium plan

B. App Service plan

C. Consumption plan

ANSWER: B

Explanation:

App Service plan is the appropriate hosting model because it can host a function app on dedicated App Service infrastructure while supporting the network integration capabilities needed for workloads that must communicate through internal virtual networks. This allows the function to access resources privately through VNet integration and enables the organization to apply its established App Service networking and security controls. A dedicated plan is also cost-effective when the organization already has available capacity in an existing App Service plan, because the function app can share that plan's allocated workers rather than requiring a separate scaling environment. This makes the incremental cost of deploying the function low while retaining predictable, dedicated compute capacity. The dedicated hosting model is especially suitable when a workload's execution, scaling, or networking needs are better served by continuously available App Service workers rather than a purely serverless consumption allocation. Azure Functions documentation describes the Dedicated (App Service) plan as running function apps on App Service plan instances, and App Service documentation describes VNet integration for outbound access to resources in a virtual network. See Azure Functions hosting and scaling and Regional VNet integration for App Service.

QUESTION NO: 35

You develop and deploy an ASP.NET web app to Azure App Service. You use Application Insights telemetry to monitor the app.

You must test the app to ensure that the app is available and responsive from various points around the world

and at regular intervals. If the app is not responding, you must send an alert to support staff.

You need to configure a test for the web app.

Which two test types can you use? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. integration

B. multi-step web

C. URL ping

D. unit

E. load

ANSWER: B C

Explanation:

multi-step web and **URL ping** are Application Insights availability test types designed to monitor an externally accessible web application on a recurring schedule from geographically distributed Azure test locations. A URL ping test sends web requests to a specified endpoint and evaluates the response, including whether the endpoint responds successfully and within the configured timeout. This provides a straightforward way to validate that the application's public URL is reachable and responsive from multiple regions.

A multi-step web test validates a recorded sequence of web interactions rather than only a single request. It is appropriate when availability depends on completing a workflow, such as navigating pages, submitting a form, or signing in. The test runs the sequence from selected locations at configured intervals and reports availability telemetry to Application Insights.

For either availability test type, an Azure Monitor alert rule can be created from the availability results. The rule can trigger when failures occur across one or more locations and can notify support staff through an action group, such as email, SMS, push notification, webhook, or an IT service-management integration. For current availability-test configuration and alerting guidance, see Application Insights availability tests and Create or edit an alert rule.

QUESTION NO: 36

You are developing a web app that is protected by Azure Web Application Firewall (WAF). All traffic to the web app is routed through an Azure Application Gateway instance that is used by multiple web apps. The web app address is contoso.azurewebsites.net.

All traffic must be secured with SSL. The Azure Application Gateway instance is used by multiple web apps.

You need to configure the Azure Application Gateway for the web app.

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

NOTE: Each correct selection is worth one point.

A. In the Azure Application Gateway's HTTP setting, enable the Use for App service setting.

B. Convert the web app to run in an Azure App service environment (ASE).

C. Add an authentication certificate for contoso.azurewebsites.net to the Azure Application Gateway.

D. In the Azure Application Gateway's HTTP setting, set the Override with new host name value to contoso.azurewebsites.net.

ANSWER: A D

Explanation:

Enable **Use for App service** in the Application Gateway HTTP setting so that the backend setting is configured appropriately for an Azure App Service multitenant endpoint. App Service uses the requested host name to identify the target site, and Application Gateway must therefore send a host header that the App Service deployment recognizes. This integration setting supports Application Gateway communication with App Service while maintaining HTTPS protection between the gateway and the backend.

Set the HTTP setting to **Override with new host name** and specify `contoso.azurewebsites.net`. The frontend listener can serve the public application hostname, but the backend request must use the App Service default hostname so that the shared App Service infrastructure routes the request to the intended web app. This host-name override is configured in the backend HTTP setting; it is not a backend-path override. Together, these settings enable secure Application Gateway-to-App-Service routing in a gateway that serves multiple web applications. For HTTPS backends, Application Gateway validates the backend certificate name against the hostname configured in the backend setting, which also makes using the App Service hostname important for the TLS connection.

See Configure App Service with Application Gateway and Application Gateway HTTP settings.

QUESTION NO: 37

You need to resolve a notification latency issue.

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

NOTE: Each correct selection is worth one point.

A. Set Always On to true.

B. Ensure that the Azure Function is using an App Service plan.

C. Set Always On to false.

D. Ensure that the Azure Function is set to use a consumption plan.

ANSWER: A B

Explanation:

Use an App Service plan and set **Always On** to true. An Azure Function hosted on a dedicated App Service plan can be unloaded after a period without activity unless Always On is enabled. Restarting the Functions host and loading the application when the anomaly webhook arrives introduces cold-start delay, which can cause the notification email to be sent several minutes after the anomaly was identified.

An App Service plan provides dedicated, continuously allocated compute for the function app. Enabling Always On sends periodic requests that keep the function app loaded, so the HTTP-triggered function can process the anomaly notification promptly rather than first waiting for the host to initialize. This setting is especially important for production function apps on dedicated App Service plans that must respond consistently to webhooks or other event-driven work. The function app must be hosted in an App Service plan that supports the Always On platform setting; then Always On should be enabled in the function app's Configuration settings.

Microsoft documents that Always On keeps an app loaded and is required for continuous execution scenarios on App Service plans. See Configure general settings for App Service and Azure Functions hosting and scaling.

QUESTION NO: 38

You have an existing Azure storage account that stores large volumes of data across multiple containers.

You need to copy all data from the existing storage account to a new storage account. The copy process must meet the following requirements:

What should you use?

A. AzCopy

B. Azure Storage Explorer

C. Azure portal

D. .NET Storage Client Library

ANSWER: A

Explanation:

AzCopy is the appropriate utility for transferring large volumes of data, including blobs and complete containers, between Azure Storage accounts. AzCopy v10 is a command-line tool optimized for high-performance, scalable data movement to, from, and between Azure Storage services. A recursive copy command can target a source container or account path and copy its contents to the destination account, making it well suited to migrating data distributed across multiple containers. It supports Microsoft Entra ID authorization and SAS tokens, so access can be granted without embedding account keys in a custom application. AzCopy also provides resumable jobs, logging, and job status commands, which are valuable operational features for long-running bulk transfers. For large migrations, its parallelized transfer engine is designed to maximize available throughput while handling the data-copy operation directly between the source and destination storage services where applicable. Microsoft documents AzCopy as the recommended command-line utility for copying blobs, directories, and containers, including copies between storage accounts. See Copy blobs between Azure storage accounts with AzCopy and Get started with AzCopy.

QUESTION NO: 39

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

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

You develop an HTTP triggered Azure Function app to process Azure Storage blob data. The app is triggered using an output binding on the blob.

The app continues to time out after four minutes. The app must process the blob data.

You need to ensure the app does not time out and processes the blob data.

Solution: Use the Durable Function async pattern to process the blob data.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: A

Explanation:

Yes. The Durable Functions asynchronous HTTP pattern is designed for operations that can outlast the HTTP request/response lifetime. Instead of retaining the original HTTP connection while blob processing runs, the HTTP-triggered starter function starts a durable orchestration and promptly returns an HTTP 202 (Accepted) response. That response includes URLs a caller can use to query orchestration status, retrieve results, or send control events. The durable orchestration then coordinates the blob-processing work independently of the initial request, typically by calling one or more activity functions.

This arrangement prevents the client-facing HTTP request from timing out while still allowing the application to complete lengthy processing reliably. Durable Functions persist orchestration state and progress in Azure Storage, enabling the workflow to resume after host restarts or transient failures rather than depending on one continuously running HTTP invocation. The processing should be implemented in activity functions, with activities kept deterministic only where appropriate and with blob content referenced or accessed efficiently rather than unnecessarily passing large payloads through orchestration history. Microsoft documents this as the HTTP asynchronous pattern for long-running operations and

describes durable orchestrations as stateful, reliable workflows. See HTTP features in Durable Functions and Durable Functions overview.

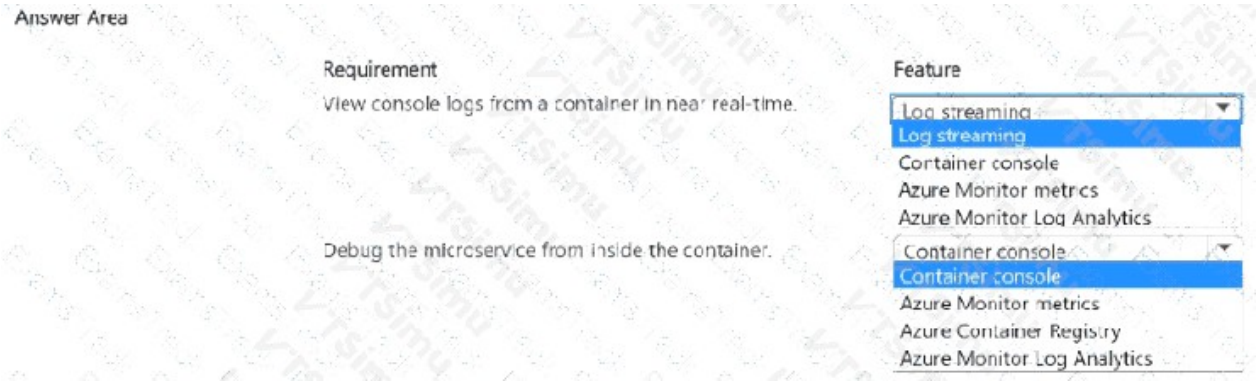
QUESTION NO: 40 - (SIMULATION)

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

You need to monitor and diagnose the microservices.

Which features should you use? To answer, select the appropriate feature in the answer area.

NOTE: Each correct selection is worth one point.



ANSWER: See the explanation for the answer

Explanation:

Select the following Azure Container Apps features:

View console logs from a container in near real-time: `Log streaming`

Debug the microservice from inside the container: `Container console`

Log streaming provides near-real-time output from the container, while Container console opens an interactive console session within the running container for troubleshooting.

QUESTION NO: 41

You are developing a complex workflow by using Azure Durable Functions.

During testing you observe that the results of the workflow differ based on how many instances of the Azure Function are running.

You need to resolve the issue.

What should you do?

A. Ensure that all Orchestrator code is deterministic.

B. Read all state data from the durable function context

C. Configure the Azure Our able f unction to run on an App Service Plan with one instance.

D. Implement the monitor pattern within the workflow.

ANSWER: A

Explanation:

Ensure that all Orchestrator code is deterministic. Durable Functions orchestrators use event sourcing: the runtime records decisions and results in a history, then replays the orchestrator code as needed to rebuild its state. Replays can occur after awaits, restarts, scale-out activity, and recovery processing. Therefore, an orchestrator must produce the same scheduling decisions every time it runs against the same orchestration history, regardless of how many function-host instances are active.

Use the Durable Functions orchestration context APIs for replay-safe operations. For example, obtain the current time through the context rather than directly from the system clock, generate identifiers through the context rather than using random or global mechanisms, and obtain external results by scheduling activities or durable entities rather than performing uncontrolled I/O in orchestration code. Keep side effects in activity functions, whose results are persisted in orchestration history and supplied consistently during replay. This deterministic design makes workflow execution reliable across scale-out and replay scenarios. Microsoft documents orchestrator replay behavior and determinism requirements in Durable Functions code constraints and provides implementation guidance in Durable Functions orchestrations.

QUESTION NO: 42

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the solution meets the stated goals.

You are developing and deploying several ASP.Net web applications to Azure App Service. You plan to save session state information and HTML output. You must use a storage mechanism with the following requirements:

• Share session state across all ASP.NET web applications

• Support controlled, concurrent access to the same session state data for multiple readers and a single writer

• Save full HTTP responses for concurrent requests

You need to store the information.

Proposed Solution: Deploy and configure Azure Cache for Redis. Update the web applications.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: A

Explanation:

Yes is correct. Azure Cache for Redis is a shared, low-latency distributed cache that can be used by multiple Azure App Service-hosted ASP.NET applications. Configuring the applications to use the Redis session-state provider moves session data out of each individual web app instance and into the shared cache. Consequently, a session remains available when requests are handled by different instances or applications that use the same configured cache.

The Redis-backed ASP.NET session-state provider supports the session locking behavior needed to coordinate access to a session. This allows the application framework to manage concurrent requests safely, including the required single-writer behavior for session updates. Redis also supports atomic operations and is designed for high-throughput concurrent client access.

Redis can additionally serve as an ASP.NET output cache. With the Redis output-cache provider configured, fully rendered HTTP responses can be cached and reused for equivalent concurrent requests, reducing repeated page generation and backend processing. The proposed deployment and application configuration therefore satisfy both the shared session-state and HTTP-response caching requirements. See Azure Cache for Redis session state provider and Azure Cache for Redis output cache provider.

QUESTION NO: 43

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

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

You are developing an Azure solution to collect point-of-sale (POS) device data from 2,000 stores located throughout the world. A single device can produce 2 megabytes (MB) of data every 24 hours. Each store location has one to five devices that send data.

You must store the device data in Azure Blob storage. Device data must be correlated based on a device identifier. Additional stores are expected to open in the future.

You need to implement a solution to receive the device data.

Solution: Provision an Azure Event Grid. Configure event filtering to evaluate the device identifier.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct. Azure Event Grid is designed for event routing: publishers emit lightweight notifications about a state change, and Event Grid delivers those events to subscribed handlers. Although Event Grid supports filtering based on event fields, including advanced filtering for values such as a device identifier, filtering does not make it an appropriate ingestion service for the POS payloads in this scenario.

Each device produces a 2-MB data payload, while Event Grid has a maximum event size of 1 MB. Therefore, the devices cannot send their full daily data as Event Grid events. A viable design would upload the payload to Azure Blob Storage and use an event notification only to communicate that a blob was created, or use an ingestion service suited to device telemetry and streaming workloads. The receiving and storage design must also preserve the device identifier, for example in blob naming, blob metadata, or the associated event data, so downstream processing can correlate data as the number of stores grows.

Microsoft documents Event Grid's event-size limit and delivery model in the Azure Event Grid quotas and limits. For guidance on selecting Azure messaging services by workload characteristics, see Choose between Azure messaging services.

QUESTION NO: 44

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

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

You are developing a website that will run as an Azure Web App. Users will authenticate by using their Azure Active Directory (Azure AD) credentials.

You plan to assign users one of the following permission levels for the website: admin, normal, and reader. A user's Azure AD group membership must be used to determine the permission level.

You need to configure authorization.

Solution:

Does the solution meet the goal?

A. Yes

B. No

ANSWER: A

Explanation:

Yes is correct. Configuring the Microsoft Entra application registration to emit group claims in authentication tokens enables the Azure Web App to obtain a signed representation of the authenticated user's group memberships. The application can then implement its authorization policy by mapping the relevant group identifiers to the required application permission levels: admin, normal, and reader. For example, after validating the token, the application can inspect its `groups` claim and grant the permissions associated with the matched group. Group claims are configured on the application registration's Token configuration settings or, where applicable, in the application manifest through the `groupMembershipClaims` setting. This approach uses Microsoft Entra ID as the authoritative source for both authentication and group membership, while the web application remains responsible for mapping groups to its own application-specific permissions. Microsoft documents how to configure group claims in tokens and explains the relevant groups claim, including considerations for group overage scenarios.

QUESTION NO: 45

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

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

You are developing an Azure solution to collect point-of-sale (POS) device data from 2,000 stores located throughout the world. A single device can produce 2 megabytes (MB) of data every 24 hours. Each store location has one to five devices that send data.

You must store the device data in Azure Blob storage. Device data must be correlated based on a device identifier. Additional stores are expected to open in the future.

You need to implement a solution to receive the device data.

Solution: Provision an Azure Event Grid. Configure the machine identifier as the partition key and enable capture.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because Azure Event Grid does not provide the ingestion features described in the proposed solution. Event Grid is an event-routing service designed to distribute discrete events from publishers to subscribers. It does not support configuring a partition key for correlated, ordered event-stream ingestion, and it does not provide the Capture feature that automatically persists an incoming event stream to Azure Blob Storage.

The stated requirements call for accepting ongoing telemetry from a growing population of POS devices, correlating records by device identifier, and retaining the received data in Blob storage. Azure Event Hubs is the Azure service that supports this model: event producers can set a partition key, such as the device or machine identifier, so events with the same key are assigned consistently to a partition. Event Hubs Capture can then automatically write the stream to Azure Blob Storage or Azure Data Lake Storage at configured time or size intervals. This combination supports scalable device ingestion while preserving partition-level ordering and grouping for each identifier.

Microsoft distinguishes Event Hubs as a high-throughput event-ingestion service from Event Grid as an event-distribution service. See Compare Azure messaging services and Event Hubs Capture overview.

QUESTION NO: 46

You develop an Azure Function app that reads messages from an Azure Service Bus queue.

The function app must authenticate to the namespace by using its system-assigned managed identity. You must not store a connection string or shared access key in the function app settings.

You need to configure access to the queue.

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

NOTE: Each correct selection is worth one point.

A. Assign the function app managed identity the Azure Service Bus Data Receiver role.

B. Configure an identity-based connection that specifies the fully qualified Service Bus namespace.

C. Store the queue shared access key in the function app code.

D. Assign the function app managed identity the Azure Service Bus Data Sender role only.

E. Enable anonymous access on the Service Bus namespace.

ANSWER: A B

Explanation:

Assign the function app's managed identity the **Azure Service Bus Data Receiver** role at the queue, namespace, or another appropriate scope. This Azure RBAC role permits the identity to receive and process messages from Service Bus entities.

Configure an **identity-based connection** by setting the connection configuration to the fully qualified Service Bus namespace, such as through the `fullyQualifiedNamespace` setting for the connection prefix. The Functions runtime can then obtain a token by using the managed identity instead of reading a connection string that contains a shared access key.

For implementation guidance, see Azure Service Bus bindings for Azure Functions and Authenticate a managed identity with Azure Service Bus.

QUESTION NO: 47

You are developing a web app that is protected by Azure Web Application Firewall (WAF). All traffic to the web app is routed through an Azure Application Gateway instance that is used by multiple web apps. The web app address is contoso.azurewebsites.net.

All traffic must be secured with SSL. The Azure Application Gateway instance is used by multiple web apps.

You need to configure the Azure Application Gateway for the app.

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

NOTE: Each correct selection is worth one point.

A. In the Azure Application Gateway's HTTP setting, enable the Use for App service setting.

B. Convert the web app to run in an Azure App service environment (ASE).

C. Add an authentication certificate for contoso.azurewebsites.net to the Azure Application gateway.

D. In the Azure Application Gateway's HTTP setting, set Override with new host name to contoso.azurewebsites.net.

ANSWER: A D

Explanation:

Enable *Use for App service* in the Application Gateway HTTP setting so that the gateway applies the App Service-specific behavior needed when it sends requests to an Azure App Service backend. This setting supports App Service backends in a shared, multitenant App Service environment and ensures the gateway uses the appropriate backend-host handling for the service. Because several web apps share the Application Gateway, the request forwarded to the backend must include the hostname that identifies the intended app.

Configure *Override with new host name* as `contoso.azurewebsites.net` in the HTTP setting. Azure App Service uses the host header to route an incoming request to the correct web app. Setting the backend host name to the app's assigned `azurewebsites.net` hostname ensures that requests forwarded by Application Gateway reach Contoso rather than another application sharing the gateway. For HTTPS backends, the configured hostname also aligns with TLS server-name indication and certificate hostname validation where applicable. The frontend listener can use HTTPS for client traffic, while the HTTP setting can be configured for HTTPS to retain encryption from Application Gateway to App Service.

For implementation details, see Configure App Service with Application Gateway and Application Gateway HTTP settings: host name.

QUESTION NO: 48

You need to grant access to the retail store location data for the inventory service development effort. What should you use?

A. Azure AD access token

B. Azure RBAC role

C. Azure AD ID token

D. Shared access signature (SAS) token

E. Azure AD refresh token

ANSWER: D

Explanation:

Shared access signature (SAS) token is the appropriate mechanism for granting the inventory service scoped, time-limited access to retail store location data held in Azure Storage. A SAS is a signed URI containing authorization parameters that specify precisely which storage resource can be accessed, which operations are permitted, and the period during which the grant is valid. For example, the service can receive permission to read only a particular blob or container containing location data, without receiving the storage account key or broader subscription-level permissions.

This supports least-privilege access and is well suited to service-to-storage data access, especially when access must be delegated for a defined duration. Azure Storage supports service SAS, account SAS, and user delegation SAS; where Microsoft Entra ID is available, a user delegation SAS is recommended because it is signed with Microsoft Entra credentials rather than an account key. SAS expiration and permissions should be limited to the minimum necessary, and tokens should be protected because anyone possessing a SAS can use the permissions it grants. See Grant limited access to Azure Storage resources using shared access signatures and SAS security best practices.

QUESTION NO: 49

You develop an application that stores orders in Azure Cosmos DB for NoSQL.

A downstream service must process each inserted or updated order asynchronously. The downstream service must be able to resume processing from its last successful position after a failure.

You need to implement the solution.

What should you use?

A. Azure Cosmos DB change feed processor

B. Azure Cosmos DB analytical store

C. Azure Cosmos DB integrated cache

D. Azure Cosmos DB transactional batch

ANSWER: A

Explanation:

Use the **Azure Cosmos DB change feed processor**. The change feed processor reads inserts and updates from a container and persists progress by using a lease container. If a processor instance stops or fails, another instance can acquire the lease and continue reading from the last checkpoint.

The change feed processor also supports distributing work across multiple consumer instances, which makes it suitable for scalable asynchronous processing of changes. Azure Cosmos DB change feed does not include deletes unless a supported all versions and deletes change feed mode is configured for the applicable scenario. For more information, see Change feed processor in Azure Cosmos DB for NoSQL.

QUESTION NO: 50

You are developing applications for a company. You plan to host the applications on Azure App Services.

The company has the following requirements:

You need to implement this process with the least amount of effort.

What should you do?

D18912E1457D5D1DDCBD40AB3BF70D5D

A. Create a Selenium web test and configure it to run from your workstation as a scheduled task.

B. Set up a URL ping test to query the home page.

C. Create an Azure function to query the home page.

D. Create a multi-step web test to query the home page.

E. Create a Custom Track Availability Test to query the home page.

ANSWER: D

Explanation:

Create a multi-step web test to query the home page. A multi-step availability test is intended to monitor a recorded user workflow rather than merely verify that a single endpoint returns a response. It can replay a sequence of HTTP requests and validate expected responses at each stage, allowing Application Insights availability monitoring to detect whether an application experience remains functional from configured test locations. This provides an Azure-managed monitoring approach: the test execution, results collection, availability telemetry, and alerting integration are handled by the platform rather than requiring the team to operate test infrastructure or write and maintain a monitoring service. For an App Service application, this is particularly useful when the required process includes navigation, authentication-independent interactions, or validation across more than one request. Results are available through Application Insights so that failed availability checks can be investigated alongside application telemetry. Microsoft documents multi-step tests as a way to monitor a recorded sequence of URLs and interactions with a web application. See Multi-step web tests in Application Insights and Availability tests and alerts in Application Insights.

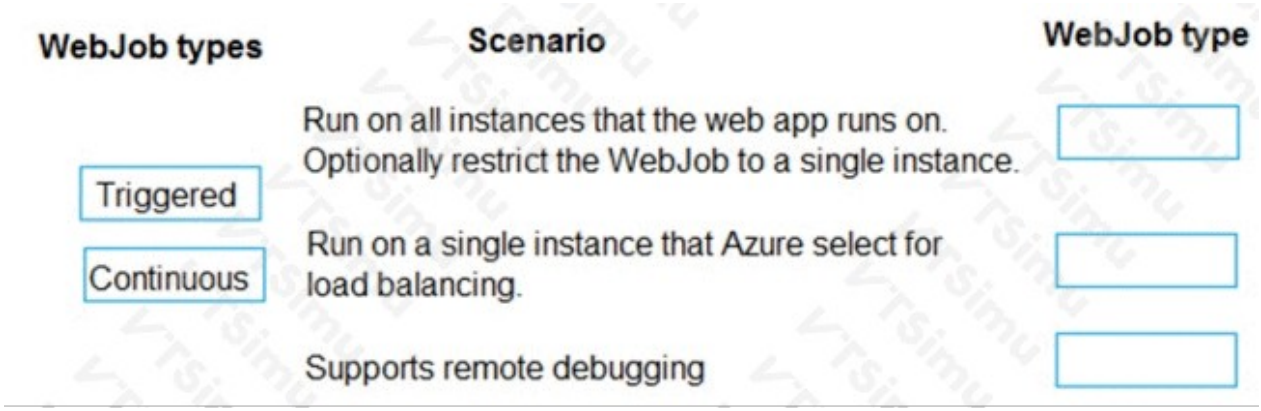
QUESTION NO: 51 - (DRAG DROP)

You are developing Azure WebJobs.

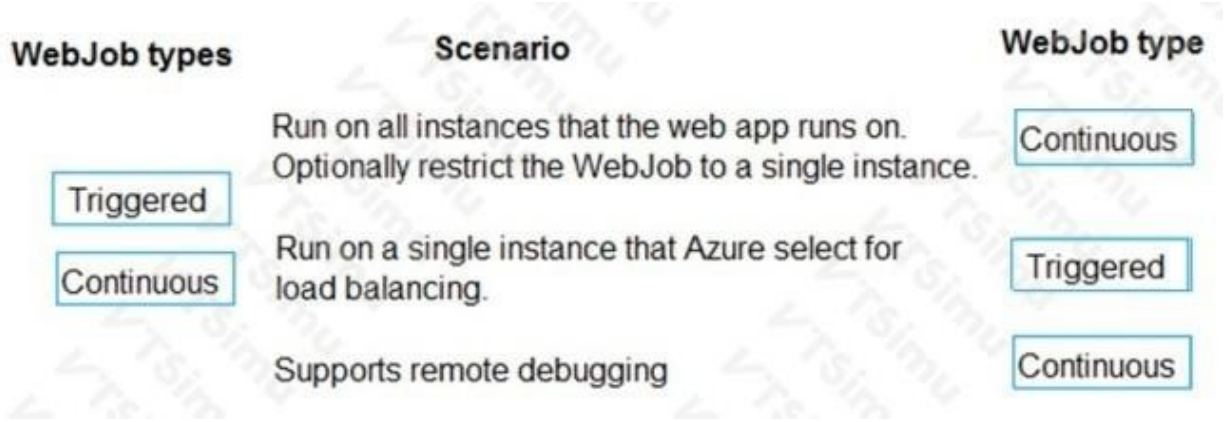
You need to recommend a WebJob type for each scenario.

Which WebJob type should you recommend? To answer, drag the appropriate WebJob types to the correct scenarios. Each WebJob type may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.



ANSWER:



Explanation:

A continuous WebJob is the appropriate recommendation when the process must run on every instance on which the App Service web app is running. This model is intended for ongoing background processing and starts automatically when the web app starts. When an app scales out, the continuous WebJob can run with the app instances. Azure also supports configuring a continuous WebJob to run on only one instance when the workload must not be duplicated. This directly satisfies the requirement to run on all instances while retaining the ability to restrict execution to a single instance.

A triggered WebJob is the right choice for a job that runs on one instance selected by Azure for load balancing. Triggered WebJobs are started manually, on a schedule, or through an external trigger, rather than remaining active continuously. Azure runs that triggered invocation on one available instance, which fits work that should execute once per invocation rather than independently on every scaled-out app instance.

Remote debugging is supported for continuous WebJobs. Because a continuous WebJob remains running as a background process, it can be attached to the remote debugger for diagnostic investigation while it executes. This is useful for long-running processing services where a developer needs to inspect runtime behavior, variables, exceptions, or execution flow without changing the WebJob model. Microsoft's WebJobs documentation describes the operating behavior of continuous and triggered jobs, including their execution and debugging characteristics. See Run background tasks with WebJobs in Azure App Service and How to use the Azure WebJobs SDK for the supported WebJobs patterns and operational details.

QUESTION NO: 52

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

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

You are developing an Azure Service application that processes queue data when it receives a message from a mobile application. Messages may not be sent to the service consistently.

You have the following requirements:

You need to implement the messaging solution.

Solution: Use the .Net API to add a message to an Azure Service Bus Queue from the mobile application. Create an Azure Windows VM that is triggered from Azure Service Bus Queue.

Does the solution meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because Azure Service Bus does not provide a native mechanism to trigger an Azure Windows virtual machine whenever a queue message arrives. A virtual machine is compute infrastructure that must be started and managed independently; an application running on it could poll or receive messages from Service Bus, but the VM itself is not an event-driven Service Bus trigger target. This does not meet the stated need for a messaging solution that processes queue data in response to messages that arrive inconsistently.

The Azure service designed for this event-driven pattern is Azure Functions. An Azure Function can use a Service Bus trigger, which monitors a queue and invokes the function when messages are available. The Functions runtime also handles message processing behavior, including scaling based on demand and completing messages after successful processing. This avoids keeping a VM running merely to wait for intermittent mobile-application messages and supports serverless, queue-driven processing. Microsoft documents Service Bus triggers as a binding that runs a function in response to messages placed on a Service Bus queue or topic. See Azure Service Bus trigger for Azure Functions and Service Bus queues, topics, and subscriptions.

QUESTION NO: 53

You need to investigate the http server log output to resolve the issue with the ContentUploadService.

Which command should you use first?

A. az webapp log tail

B. az ams live-output

C. az monitor activity-log

D. az container attach

ANSWER: A

Explanation:

`az webapp log tail` is the appropriate first command because it streams the application's live log output from the Azure App Service hosting ContentUploadService. For intermittent HTTP 502 responses, observing the log stream while reproducing the affected request can reveal request failures, application exceptions, startup failures, upstream connectivity events, and other runtime details correlated with the error time. This directly supports initial investigation of HTTP server

behavior rather than examining management-plane events. App Service logging must be enabled for the applicable log sources, such as application logging or web server logging, so that relevant entries are available in the stream. If logging has not yet been configured, configure the required App Service logging settings first and then use live log tailing to collect current diagnostic evidence. Microsoft documents `az webapp log tail` as the Azure CLI command for starting live log tracing for an App Service app. See the Azure CLI reference for az webapp log tail and Enable diagnostics logging for Azure App Service.

QUESTION NO: 54

You are developing an Azure App Service REST API.

The API must be called by an Azure App Service web app. The API must retrieve and update user profile information stored in Azure Active Directory (Azure AD).

You need to configure the API to make the updates.

Which two tools should you use? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Microsoft Graph API

B. Microsoft Authentication Library (MSAL)

C. Azure API Management

D. Microsoft Azure Security Center

E. Microsoft Azure Key Vault SDK

ANSWER: A B

Explanation:

Microsoft Graph API is the Microsoft-supported REST API for accessing Microsoft Entra ID (formerly Azure AD) directory resources, including user objects and supported profile properties. The API can retrieve user information through endpoints such as `/users/{id}` and can update permitted user properties with a `PATCH` request. The application must be registered in Microsoft Entra ID and granted the appropriate delegated or application permissions, such as `User.ReadWrite.All`, with administrator consent where required.

Microsoft Authentication Library (MSAL) supplies the authentication and token-acquisition functionality that the App Service API needs before calling Microsoft Graph. In this web-app-to-API design, MSAL supports obtaining and validating Microsoft Entra ID tokens and acquiring a Graph access token using the appropriate flow, such as the on-behalf-of flow when the API performs Graph operations for the signed-in user, or a confidential-client flow for app-only operations. The resulting access token is sent in the Authorization header of Microsoft Graph requests. Together, these tools provide both the authorized directory API surface and the OAuth token mechanism required to perform profile updates securely.

See Update user with Microsoft Graph and Microsoft Authentication Library overview.

QUESTION NO: 55

You develop a REST API. You implement a user delegation SAS token to communicate with Azure Blob

storage.

The token is compromised.

You need to revoke the token.

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

NOTE: Each correct selection is worth one point.

A. Revoke the delegation keys

B. Delete the stored access policy.

C. Regenerate the account key.

D. Remove the role assignment for the security principle.

E. Wait for the user delegation SAS or the user delegation key to expire.

ANSWER: A E

Explanation:

For a user delegation SAS, **Revoke the delegation keys** is the direct, immediate revocation mechanism. A user delegation SAS is signed with a user delegation key that Microsoft Entra ID authorizes for the storage account. Revoking the account's user delegation keys invalidates user delegation SAS tokens that were signed with those keys. Azure documents this operation through the `az storage account revoke-delegation-keys` command and equivalent management operations. This is the appropriate response when a compromised token must be invalidated before its configured expiry time.

Wait for the user delegation SAS or the user delegation key to expire also results in the compromised token becoming unusable. User delegation SAS access is bounded by both the SAS expiry and the lifetime of the user delegation key. Azure Storage will no longer authorize a request after the SAS expiry time or after the corresponding user delegation key expires, whichever occurs first. User delegation keys have a maximum validity period of seven days, so expiration provides a bounded automatic invalidation path when immediate revocation is not required. For details, see Create a user delegation SAS and Revoke User Delegation Keys.