KillTest

質量更高　服務更好

學習資料

http://www.killtest.net

一年免費更新服務

Exam : **AZ-203**

Title : Developing Solutions for Microsoft Azure

Version : DEMO

1. Testlet 1

Case Study
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.

To start the case study
To display the first question on this case study, click the **Next** button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an **All Information** tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a **question**, click the Question button to return to the question.

Background
You are a developer for Proseware, Inc. You are developing an application that applies a set of governance policies for Proseware's internal services, external services, and applications. The application will also provide a shared library for common functionality.

Requirements
Policy service
You develop and deploy a stateful ASP.NET Core 2.1 web application named Policy service to an Azure App Service Web App. The application reacts to events from Azure Event Grid and performs policy actions based on those events.
The application must include the Event Grid Event ID field in all Application Insights telemetry.
Policy service must use Application Insights to automatically scale with the number of policy actions that it is performing.

Policies
Log Policy
All Azure App Service Web Apps must write logs to Azure Blob storage. All log files should be saved to a container named **logdrop**. Logs must remain in the container for 15 days.

Authentication events
Authentication events are used to monitor users signing in and signing out. All authentication events must be processed by Policy service. Sign outs must be processed as quickly as possible.

PolicyLib
You have a shared library named **PolicyLib** that contains functionality common to all ASP.NET Core web services and applications.
The **PolicyLib** library must:
- Exclude non-user actions from Application Insights telemetry.
- Provide methods that allow a web service to scale itself
- Ensure that scaling actions do not disrupt application usage

Other
Anomaly detection service
You have an anomaly detection service that analyzes log information for anomalies. It is implemented as an Azure Machine Learning model. The model is deployed as a web service.
If an anomaly is detected, an Azure Function that emails administrators is called by using an HTTP WebHook.

Health monitoring
All web applications and services have health monitoring at the /health service endpoint.

Issues
Policy loss
When you deploy Policy service, policies may not be applied if they were in the process of being applied during the deployment.

Performance issue
When under heavy load, the anomaly detection service undergoes slowdowns and rejects connections.

Notification latency
Users report that anomaly detection emails can sometimes arrive several minutes after an anomaly is detected.

App code
Relevant portions of the app files are shown below. Line numbers are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

EventGridController.cs

```
EG01  public class EventGridController : Controller
EG02  {
EG03    public static AsyncLocal<string> EventId = new AsyncLocal<string>();
EG04    public IActionResult Process([FromBody]) string eventsJson
EG05   {
EG06     var events = JArray.Parse(eventsJson);
EG07
EG08     foreach (var @event in events)
EG09     {
EG10       EventId.Value = @event ["id"].ToString();
EG11       if (@event["topic"].ToString().Contains("providers/Microsoft.Storage"))
EG12       {
EG13         SendToAnomalyDetectionService(@event["data"]["url"].ToString());
EG14        }
EG15
EG16        {
EG17          EnsureLogging(@event["subject"].ToString());
EG18         }
EG19       }
EG20        return null;
EG21   }
EG22   private void EnsureLogging(string resource)
EG23   {
EG24     . . .
EG25   }
EG26   private async Task SendToAnomalyDetectionService(string uri)
EG27   {
EG28     var content = GetLogData(uri);
EG29     var scoreRequest = new
EG30     {
EG31       Inputs = new Dictionary<string, List<Dictionary<string, string>>>()
EG32       {
EG33          {
EG34            "input1",
```

```
EG35            new List<Dictionary<string, string>>()
EG36            {
EG37               new Dictionary<string, string>()
EG38               {
EG39             {
EG40               "logcontent", content
EG41              }
EG42             }
EG43           }
EG44         },
EG45       },
EG46       GlobalParameters = new Dictionary<string, string>() { }
EG47      };
EG48      var result = await (new HttpClient()).PostAsJsonAsync(". . .", scoreRequest);
EG49      var rawModelResult = await result.Content.ReadAsStringAsync();
EG50      var modelResult = JObject.Parse(rawModelResult);
EG51      if (modelResult["notify"].HasValues)
EG52      {
EG53        . . .
EG54       }
EG55    }
EG56    private (string name, string resourceGroup) ParseResourceId(string
resourceId)
EG57    {
EG58      . . .
EG59     }
EG60     private string GetLogData(string uri)
EG61    {
EG62      . . .
EG63    }
EG64    static string BlobStoreAccountSAS(string containerName)
EG65    {
EG66      . . .
EG67     ]
EG68  }
```

Relevant portions of the app files are shown below.
Line numbers are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

LoginEvent.cs

```
LE01  public class LoginEvent
LE02  {
LE03
LE04  public string subject { get; set; }
LE05  public DateTime eventTime { get; set; }
LE06  public Dictionary<string, string> data { get; set; }
LE07  public string Serialize()
LE08  {
LE09    return JsonConvert.SerializeObject(this);
LE10   }
LE11 }
```

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 false.
B. Set Always On to true.
C. Ensure that the Azure Function is set to use a consumption plan.
D. Ensure that the Azure Function is using an App Service plan.
Answer: BD
Explanation:
Azure Functions can run on either a Consumption Plan or a dedicated App Service Plan. If you run in a dedicated mode, you need to turn on the Always On setting for your Function App to run properly. The Function runtime will go idle after a few minutes of inactivity, so only HTTP triggers will actually "wake up" your functions. This is similar to how WebJobs must have Always On enabled.
Scenario: Notification latency: Users report that anomaly detection emails can sometimes arrive several minutes after an anomaly is detected.
Anomaly detection service: You have an anomaly detection service that analyzes log information for anomalies. It is implemented as an Azure Machine Learning model. The model is deployed as a web service.
If an anomaly is detected, an Azure Function that emails administrators is called by using an HTTP WebHook.
References:
https://github.com/Azure/Azure-Functions/wiki/Enable-Always-On-when-running-on-dedicated-App-Service-Plan

2. Testlet 2

Case Study
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.

To start the case study
To display the first question on this case study, click the **Next** button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case

study has an **All Information** tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a **question**, click the Question button to return to the question.

LabelMaker app
Coho Winery produces bottles, and distributes a variety of wines globally. You are a developer implementing highly scalable and resilient applications to support online order processing by using Azure solutions.

Coho Winery has a LabelMaker application that prints labels for wine bottles. The application sends **data** to several printers. The application consists of five modules that run independently on virtual machines (VMs). Coho Winery plans to move the application to Azure and continue to support label creation.

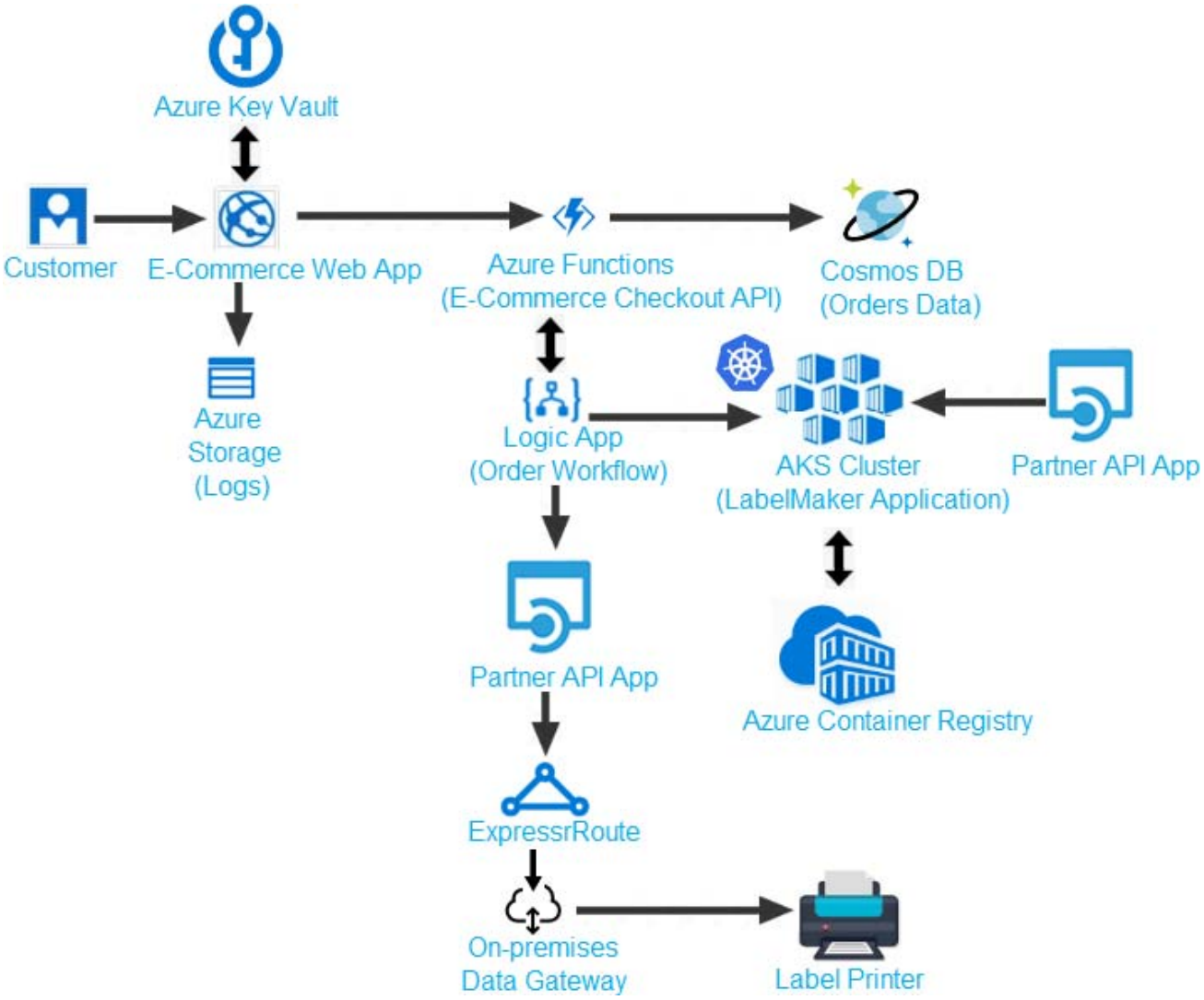
External partners send data to the **LabelMaker app**lication to include artwork and text for custom label designs.

Requirements
Data
You identify the following requirements for data management and manipulation:
• Order data is stored as nonrelational JSON and must be queried using Structured Query Language (SQL).
• Changes to the Order data must reflect immediately across all partitions. All reads to the Order data must fetch the most recent writes.

Security
You have the following security requirements:
• Users of Coho Winery applications must be able to provide access to documents, resources, and applications to external partners.
• External partners must use their own credentials and authenticate with their organization's identity management solution.
• External partner logins must be audited monthly for application use by a user account administrator to maintain company compliance.
• Storage of e-commerce application settings must be maintained in Azure Key Vault.
• E-commerce application sign-ins must be secured by using Azure App Service authentication and Azure Active Directory (AAD).
• Conditional access policies must be applied at the application level to protect company content.
• The LabelMaker application must be secured by using an AAD account that has full access to all namespaces of the Azure Kubernetes Service (AKS) cluster.

LabelMaker app
Azure Monitor Container Health must be used to monitor the performance of workloads that are deployed to Kubernetes environments and hosted on Azure Kubernetes Service (AKS).
You must use Azure Container Registry to publish images that support the AKS deployment.

Architecture



Issues

Calls to the Printer API App fail periodically due to printer communication timeouts.

Printer communications timeouts occur after 10 seconds. The label printer must only receive up to 5 attempts within one minute.

The order workflow fails to run upon initial deployment to Azure.

Order .json

Relevant portions of the app files are shown below. Line numbers are included for reference only.

This JSON file contains a representation of the data for an order that includes a single item.

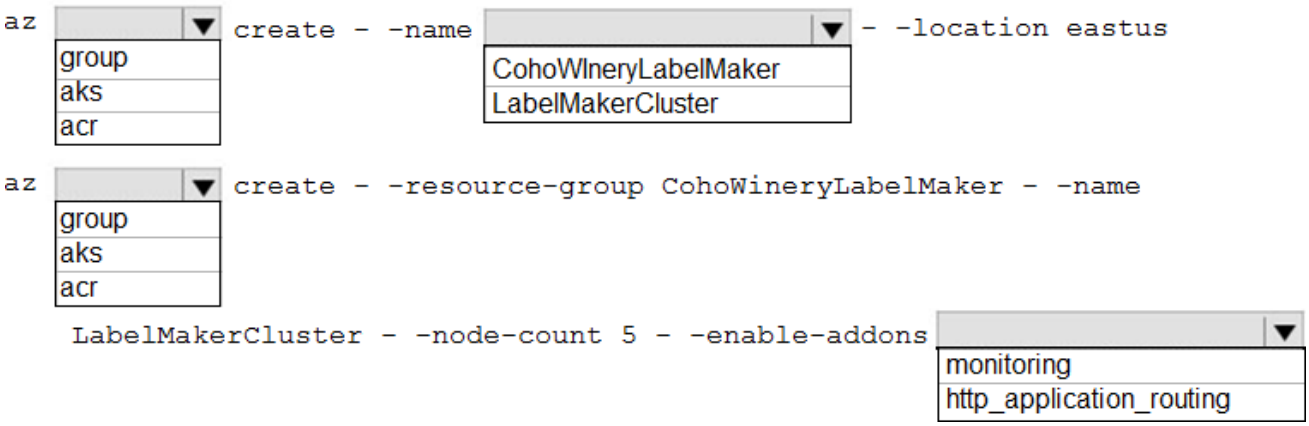
```
{
 "id : 1,
 "customers" : [
 {
   "familyName" : "Doe",
   "givenName" :  "John",
   "customerid" :  5
 }
 ],
 "line"_items" : [
{
  "fulfillable_quantity" : 1,
 "id" : 6,
 "price" : "199.99",
 "product_id" : 7513594,
 "quantity": 1,
 "requires_shipping" : true,
 "sku" : "SFC-342-N" ,
"title" : "Surface Go",
 "vendor" : "Microsoft" ,
 "name" : "Surface Go - 8GB",
 "taxable" : true,
 "tax"_lines" : [
 {
  "title" : "State Tax",
 "price" : "3.98",
 "rate" : 0.06
 }
],
 "total_discount" : "5.00"
 "discount_allocations" : [
 {
  "amount" : "5.00",
  "discount_application_index" : 2
    }
   ]
  }
 ],
"address" : {
"state" : "NY",
"country" : "Manhattan",
 "city" : "NY"
   }
  }
```

HOTSPOT

You need to ensure that you can deploy the LabelMaker application.

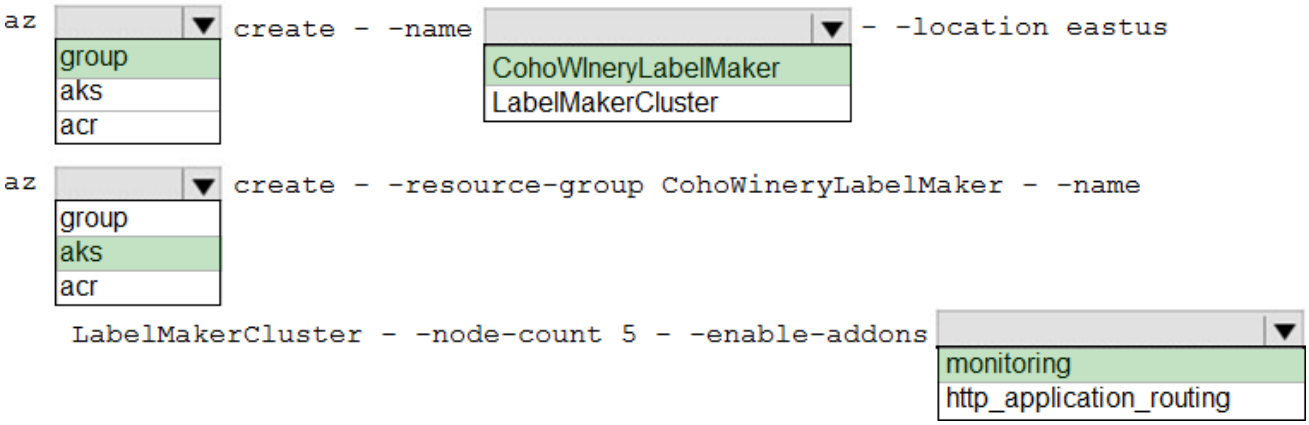
How should you complete the CLI commands? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Explanation:

Box 1: group

Create a resource group with the az group create command. An Azure resource group is a logical group in which Azure resources are deployed and managed.

The following example creates a resource group named myResourceGroup in the westeurope location.

az group create --name myResourceGroup --location westeurope

Box 2: CohoWinterLabelMaker

Use the resource group named, which is used in the second command.

Box 3: aks

The command az aks create, is used to create a new managed Kubernetes cluster.

Box 4: monitoring

Scenario: LabelMaker app

Azure Monitor Container Health must be used to monitor the performance of workloads that are deployed to Kubernetes environments and hosted on Azure Kubernetes Service (AKS).

You must use Azure Container Registry to publish images that sup

3. Testlet 3

Case Study
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.

To start the case study
To display the first question on this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Background
Wide World Importers is moving all their datacenters to Azure. The company has developed several applications and services to support supply chain operations and would like to leverage serverless computing where possible.

Current environment
Windows Server 2016 virtual machine
This virtual machine (VM) runs Biz Talk Server 2016.
The VM runs the following workflows:
- Ocean Transport – This workflow gathers and validates container information including container contents and arrival notices at various shipping ports.
- Inland Transport – This workflow gathers and validates trucking information including fuel usage, number of stops, and routes.

The VM supports the following REST API calls:
- Container API – This API provides container information including weight, contents, and other attributes.
- Location API – This API provides location information regarding shipping ports of call and truck stops.

- Shipping REST API – This API provides shipping information for use and display on the shipping website.

Shipping Data

The application uses MongoDB JSON document storage database for all container and transport information.

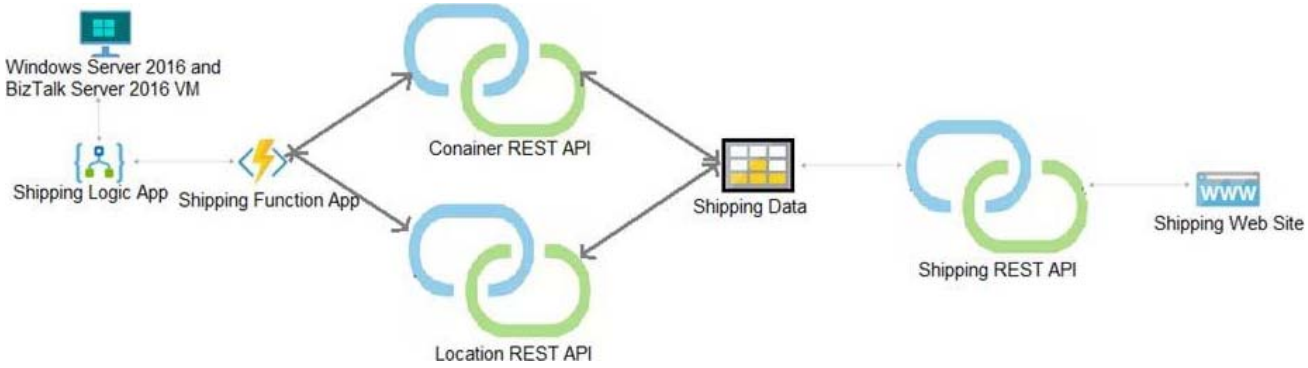
Shipping Web Site

The site displays shipping container tracking information and container contents. The site is located at http://shipping.wideworldimporters.com

Proposed solution

The on-premises shipping application must be moved to Azure. The VM has been migrated to a new Standard_D16s_v3 Azure VM by using Azure Site Recovery and must remain running in Azure to complete the BizTalk component migrations. You create a Standard_D16s_v3 Azure VM to host BizTalk Server.

The Azure architecture diagram for the proposed solution is shown below:



Shipping Logic App

The Shipping Logic app must meet the following requirements:

- Support the ocean transport and inland transport workflows by using a Logic App.
- Support industry-standard protocol X12 message format for various messages including vessel content details and arrival notices.
- Secure resources to the corporate VNet and use dedicated storage resources with a fixed costing model.
- Maintain on-premises connectivity to support legacy applications and final BizTalk migrations.

Shipping Function app

Implement secure function endpoints by using app-level security and include Azure Active Directory (Azure AD).

REST APIs

The REST API's that support the solution must meet the following requirements:

- Secure resources to the corporate VNet.
- Allow deployment to a testing location within Azure while not incurring additional costs.

- Automatically scale to double capacity during peak shipping times while not causing application downtime.
- Minimize costs when selecting an Azure payment model.

Shipping data
Data migration from on-premises to Azure must minimize costs and downtime.

Shipping website
Use Azure Content Delivery Network (CDN) and ensure maximum performance for dynamic content while minimizing latency and costs.

Issues
Windows Server 2016 VM
The VM shows high network latency, jitter, and high CPU utilization. The VM is critical and has not been backed up in the past. The VM must enable a quick restore from a 7-day snapshot to include in-place restore of disks in case of failure.

Shipping website and REST APIs
The following error message displays while you are testing the website:

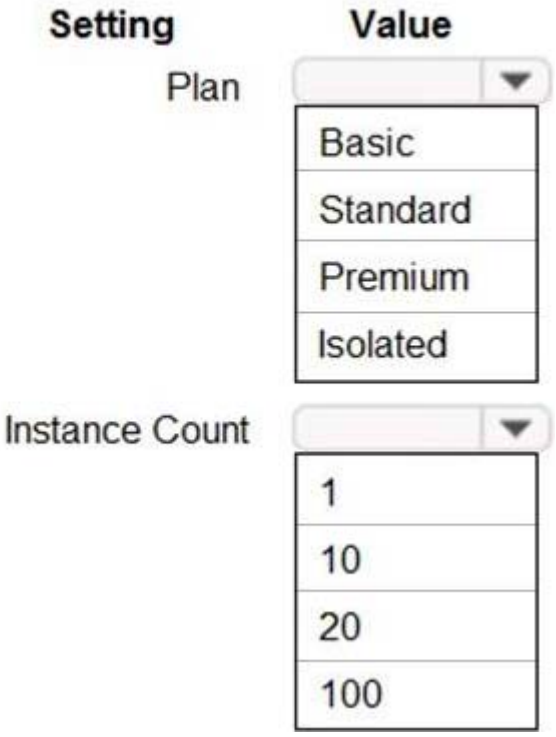
```
Failed to load http://test-shippingapi.wideworldimporters.com/: No 'Access-Control-Allow-Origin'
header is present on the requested resource. Origin 'http://testwideworldimporters.com/' is
therefore not allowed access.
```

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:
Before you can connect to on-premises data sources from Azure Logic Apps, download and install the on-premises data gateway on a local computer. The gateway works as a bridge that provides quick data transfer and encryption between data sources on premises (not in the cloud) and your logic apps.
The gateway supports BizTalk Server 2016.
Note: Microsoft have now fully incorporated the Azure BizTalk Services capabilities into Logic Apps and Azure App Service Hybrid Connections.
Logic Apps Enterprise Integration pack bring some of the enterprise B2B capabilities like AS2 and X12, EDI standards support
Scenario: The Shipping Logic app must meet the following requirements:
- Support the ocean transport and inland transport workflows by using a Logic App.
- Support industry standard protocol X12 message format for various messages including vessel content details and arrival notices.
- Secure resources to the corporate VNet and use dedicated storage resources with a fixed costing

model.
- Maintain on-premises connectivity to support legacy applications and final BizTalk migrations.
References: https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-gateway-install

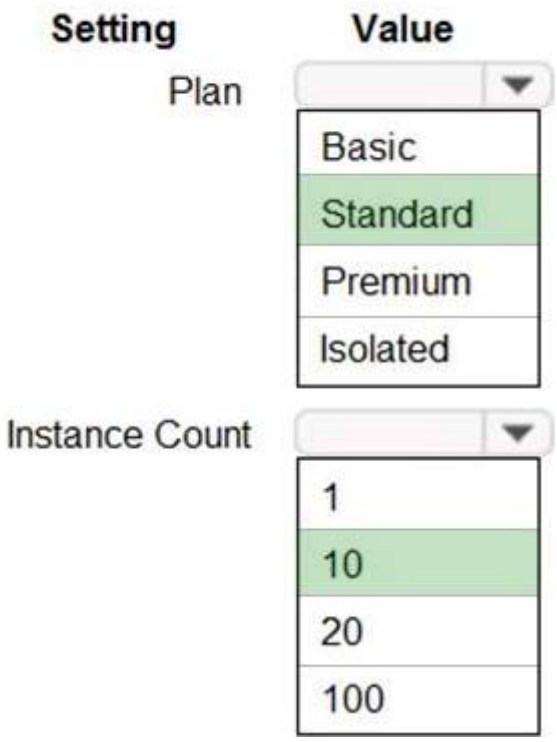
4.HOTSPOT
You need to configure Azure App Service to support the REST API requirements.
Which values should you use? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Explanation:
Plan: Standard
Standard support auto-scaling
Instance Count: 10
Max instances for standard is 10.
Scenario:
The REST API's that support the solution must meet the following requirements:
- Allow deployment to a testing location within Azure while not incurring additional costs.
- Automatically scale to double capacity during peak shipping times while not causing application downtime.
- Minimize costs when selecting an Azure payment model.
References: https://azure.microsoft.com/en-us/pricing/details/app-service/plans/

5. Question Set 4

You are writing code to create and run an Azure Batch job. You have created a pool of compute nodes.
You need to choose the right class and its method to submit a batch job to the Batch service.
Which method should you use?
A. JobOperations.EnableJobAsync(String, IEnumerable<BatchClientBehavior>,CancellationToken)
B. JobOperations.CreateJob()
C. CloudJob.Enable(IEnumerable<BatchClientBehavior>)
D. JobOperations.EnableJob(String,IEnumerable<BatchClientBehavior>)
E. CloudJob.CommitAsync(IEnumerable<BatchClientBehavior>, CancellationToken)

Answer: E

Explanation:

A Batch job is a logical grouping of one or more tasks. A job includes settings common to the tasks, such as priority and the pool to run tasks on. The app uses the BatchClient.JobOperations.CreateJob method to create a job on your pool.

The Commit method submits the job to the Batch service. Initially the job has no tasks.

```
{
    CloudJob job = batchClient.JobOperations.CreateJob();
    job.Id = JobId;
    job.PoolInformation = new PoolInformation { PoolId = PoolId };

    job.Commit();
}
...
```

References: https://docs.microsoft.com/en-us/azure/batch/quick-run-dotnet