```
"print_label": {
  "type": "Http",
  "inputs": {
    "method" : "POST",
    "uri": "https://www.cohowinery.com/printer/printlabel",
    "retriyPolicy": {
    "type": "  [▼]  ",
               default
               none
               fixed
               exponential
    "interval": "  [▼]  ",
                   PT10S
                   PT30S
                   PT60S
                   PT1D
     "count":  [▼]
               5
               10
               60
      }
    }
  }
```

QUESTION 15
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.

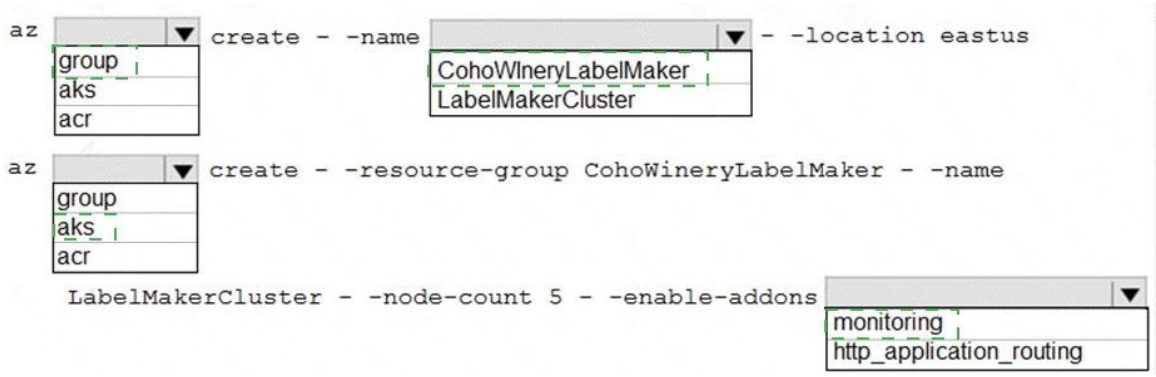
NOTE: Each correct selection is worth one point.

```
az [▼] create - -name [▼] - -location eastus
   group              CohoWIneryLabelMaker
   aks                LabelMakerCluster
   acr

az [▼] create - -resource-group CohoWineryLabelMaker - -name
   group
   aks
   acr

  LabelMakerCluster - -node-count 5 - -enable-addons [▼]
                                                     monitoring
                                                     http_application_routing
```

Correct Answer:



QUESTION 16
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 need to meet the LabelMaket application

Solution; Create a conditional access policy and assign it to the Azure Kubernetes service cluster.

Does the solution meet the goal?

A. Yes
B. No

Correct Answer: B
Explanation:
Scenario: The LabelMaker applications must be secured by using an AAD account that has full access to all namespaces of the Azure Kubernetes Service (AKS) cluster.

Before an Azure Active Directory account can be used with the AKS cluster, a role binding or cluster role binding needs to be created.

References:
https://docs.microsoft.com/en-us/azure/aks/aad-integration

Topic 2, Litware Inc

Overview
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 infonnation 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 questions in 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 section 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 in this case study, dick 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.

Overwiew
Background
You are a developer for Litware Inc., a SaaS company that provides a solution for managing employee expenses. The solution consists of an ASP.NET Core Web API project that is deployed as an Azure Web App.

Overall architecture
Employees upload receipts for the system to process. When processing is complete, the employee receives a summary report email that details the processing results. Employees then use a web application to manage their receipts and perform any additional tasks needed for reimbursement

Receipt processing
Employees may upload receipts in two ways:

- Uploading using an Azure Files mounted folder
- Uploading using the web application

Data Storage
Receipt and employee information is stored in an Azure SQL database.

Documentation
Employees are provided with a getting started document when they first use the solution. The documentation includes details on supported operating systems for Azure File upload, and instructions on how to configure the mounted folder.

Solution details
Users table

Column	Description
UserId	unique identifier for and employee
ExpenseAccount	employees expense account number in the format 1234-123-1234
AllowedAmount	limit of allowed expenses before approval is needed
SupervisorId	unique identifier for employee's supervisor
SecurityPin	value used to validate user identity

Web Application
You enable MSI for the Web App and configure the Web App to use the security principal name,

Processing
Processing is performed by an Azure Function that uses version 2 of the Azure Function runtime. Once processing is completed, results are stored in Azure Blob. Storage and an Azure SQL database. Then, an email summary is sent to the user with a link to the processing report. The link to the report must remain valid if the email is forwarded to another user.

Requirements
Receipt processing
Concurrent processing of a receipt must be prevented.

Logging
Azure Application Insights is used for telemetry and logging in both the processor and the web application. The processor also has Trace Writer logging enabled. Application Insights must always contain all log messages.

Disaster recovery
Regional outage must not impact application availability. All DR operations must not be dependent on application running and must ensure that data in the DR region is up to date.

Security
Users' SecurityPin must be stored in such a way that access to the database does not allow the viewing of SecurityPins. The web application is the only system that should have access to SecurityPins.

All certificates and secrets used to secure data must be stored in Azure Key Vault.

You must adhere to the Least Privilege Principal.

All access to Azure Storage and Azure SQL database must use the application's Managed Service Identity (MSI).

Receipt data must always be encrypted at rest.

All data must be protected in transit,

User's expense account number must be visible only to logged in users. All other views of the expense account number should include only the last segment, with the remaining parts obscured.

In the case of a security breach, access to all summary reports must be revoked without impacting other parts of the system.

Issues

Upload format issue
Employees occasionally report an issue with uploading a receipt using the web application. They report that when they upload a receipt using the Azure File Share, the receipt does not appear in their profile. When this occurs, they delete the file in the file share and use the web application, which returns a 500 Internal Server error page.

Capacity issue
During busy periods, employees report long delays between the time they upload the receipt and when it appears in the web application.

Log capacity issue
Developers report that the number of log messages in the trace output for the processor is too high, resulting in lost log messages.

Application code
Processing.cs
Processing.cs

```
PC01 public static class Processing
PC02  {
PC03     public static class Function
PC04     {
PC05       [FunctionName ("IssueWork")]
PC06       public static async Task Run ([TimerTrigger("0 "/5" *****")] TimerInfo timer, ILogger log)
PC07       {
PC08         var container = await GetCloudBlobContainer();
PC09         foreach (var fileItem in await ListFiles())
PC10         {
PC11           var file = new CloudFile (fileItem.StorageUri.PrimaryUri);
PC12           var ms = new MemoryStream();
PC13           await file.DownloadToStreamAsync(ms);
PC14           var blob = container.GetBlockBlobReference (fileItem.Uri.ToString());
PC15           await blob.UploadFromStreamAsync(ms);
PC16
PC17         }
PC18   }
PC19   private static CloudBlockBlob GetDRBlob (CloudBlockBlob sourceBlob)
PC20   {
PC21     . . .
PC22   }
PC23   private static async Task<CloudBlobContainer> GetCloudBlobContainer()
PC24   {
PC25     var cloudBlobClient = new CloudBlobClient (new Uri(" . . ."), await GetCredentials());
PC26
PC27    await cloudBlobClient.GetRootContainerReference().CreatIfNotExistAsync();
PC28    return cloudBlobClient.GetRootContainerReference();
PC29  }
PC30   private static async Task<StorageCredentials> GetCredentials()
PC31   {
PC32      . . .
PC33   }
PC34    private static async Task<List<IListFileItem>> ListFiles()
PC35   {
PC36      . . .
PC37   }
PC37   private KeyVaultClient _keyVaultClient = new KeyVaultClient(". . .");
PC38   }
PC39 }
```

Database.cs