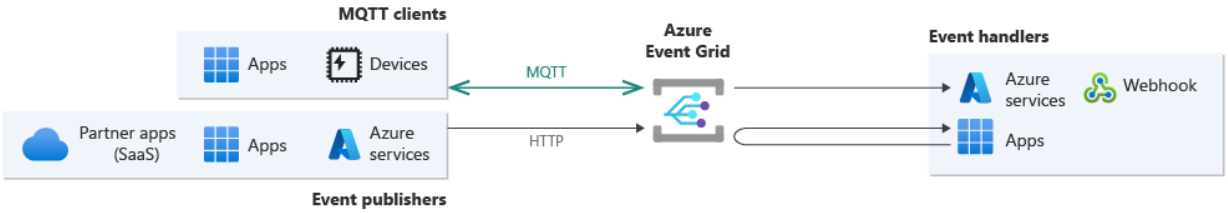
iPub-Que-Sub: Grids, Hubs, Events [logical boundaries & crossing them=?]

1. Events
    a. Used for ephemeral broadcasts - communication might not be handled by any receiver.
    b. Pub/Sub
    c. Managed by an intermediary, like Azure Event Grid or Azure Event Hubs. When publishers send an event, the intermediary will route that event to interested subscribers.
    d.

```
[
  {
    "topic": string,
    "subject": string,
    "id": string,
    "eventType": string,
    "eventTime": string,
    "data":{
      object-unique-to-each-publisher
    },
    "dataVersion": string,
    "metadataVersion": string
  }
]
```

Events are the data messages describe what has taken place.

2. Messages [JSON]
    a. Used where the distributed application requires a guarantee that the communication will be processed.
3. Event Grid [namespace & topics]
    a. Event routing service running on top of Azure Service Fabric that Distributes events from different sources, such as Azure Blob storage accounts or Azure Media Services, to different handlers, such as Functions or Webhooks.
    b.



    c. Pub Sub message distribution service
    d. Use topics to decide which events to send to which handlers
    e.

concepts in Azure Event Grid

- **Events**: What happened.
- **Event sources**: Where the event took place.
- **Topics**: The endpoint where publishers send events.
- **Event subscriptions**: The endpoint or built-in mechanism to route events, sometimes to multiple handlers. Subscriptions are also used by handlers to filter incoming events intelligently.
- **Event handlers**: The app or service reacting to the event.

    f. [Publish and consume events or messages using namespace topics - Azure Event Grid | Microsoft Learn]
4. Event Topic

   a. Event topics categorize events into groups.
5. Event Hub: support sales and marketing team to *analyze live user* behavior.
   a. Real time Telemetry and distributed data streaming
   b. Namespace is the parent container for managing one or more Event Hubs and when to use more than one namespace
      i. One for data one for logs.
      ii. namespace.servicebus.windows.net
      iii.
   c. Azure Event Hubs is a big data pipeline.
   d. It facilitates the capture, retention, and replay of telemetry and event stream data. The data can come from many concurrent sources.
   e. Event Hubs allows telemetry and event data to be made available to a variety of stream-processing infrastructures and analytics services.
   f. It is available either as data streams or bundled event batches.
   g. This service provides a single solution that enables rapid data retrieval for real-time processing
   h. as well as repeated replay of stored raw data.

   The following scenarios are some of the scenarios where you can use Event Hubs:

   i. Anomaly detection (fraud/outliers)
   j. Application logging
   k. Analytics pipelines, such as clickstreams
   l. Live dashboarding
   m. Archiving data
   n. Transaction processing
   o. User telemetry processing
   p. Device telemetry streaming
   q. https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about
6. Event Hub Consumer Group
   a. Represents a specific view of an Event Hub data stream
7. Compare Azure Storage queues and Service Bus queues - Azure Service Bus | Microsoft Learn
8. Storage Queue
   a. Queues are commonly used to create a backlog of work to process asynchronously.
   b. You need an audit trail of all messages that pass through the queue.
   c. You expect the queue to exceed 80 GB in size.
   d. You need to handle messages less than 64 KB.
   e. You want to track progress for processing a message inside of the queue

Service Bus Components: order processing - applications require transactions, ordering, duplicate detection, and instantaneous consistency.

9. Service Bus Queue
   a. You need an At-Most-Once delivery guarantee.
   b. You need a **FIFO** guarantee.

   c. You need to group messages into **transactions**.
   d. You want to receive messages **without polling** the queue.
   e. You need to provide a **role-based access** model to the queues.
   f. You need to handle messages larger than 64 KB but less than 256 KB.
   g. Your queue size will not grow larger than 80 GB.
   h. You would like to be able to publish and consume batches of messages.
10. Service Bus Topics
    a. You need multiple subscribers to do the get messages.
    b. When you post to a topic, the message is copied and dropped into the queue
       i. for each subscription.
       ii. The message copy will stay around to be processed by each subscription branch.
11. Service Fabric is an open-source project and it powers core Azure infrastructure as well as other Microsoft services such as Skype for Business, Intune, Azure Event Hubs, Azure Data Factory, Azure Cosmos DB, Azure SQL Database, Dynamics 365, and Cortana.
12. Azure Notification Hubs
13. Space

Trigger something – do something – what are somethings todo?

1. Authentication events are triggered and processed according to the authentication events policy.
2. information should you cache?
3. send a notification to user's device when there is a new article available for them to view.
4. AAD account that has full access to all namespaces of the Azure Kubernetes Service (AKS) cluster.
5. access data from the user claim object
6. API app is expected to allow users to authenticate by using Twitter and Azure Active Directory (Azure AD) credentials.
7. log the users name for each API call.
8. copy specific blobs from Container1 to Container2 in real time when specific business requirements are met.
9. send email alerts to warehouse supervisors immediately if the temperature at a warehouse goes above or below specified threshold temperatures.
10. Files must be purged from the FrontDoor cache based upon Response Header values.
11. Implement a reply trail auditing solution.
12. *Create an Azure Function and configure it to deploy code from a GitHub repository.
13. *Azure Event Grid must use Azure Service Bus for queue-based load leveling. Events in Azure Event Grid must be routed directly to Service Bus queues for use in buffering.
    a. Events from Azure Service Bus and other Azure services must continue to be routed to Azure Event Grid for processing.
    b. Use a queue that acts as a buffer between a task and a service it invokes in order to smooth intermittent heavy loads that can cause the service to fail or the task to time out.
14. Manage artifact metadata in integration accounts - Azure Logic Apps | Microsoft Learn
15. Azure Batch pool

AzCopy

[Find errors & resume jobs with logs in AzCopy (Azure Storage) | Microsoft Learn]

Azure CDN rule.
How should you complete the Azure Resource Manager template?

```
"conditions": [ {
    "name": "IsDevice",
    "parameters": {
      "@odata.type": "#Microsoft.Azure.Cdn.Models. <CodeBlock1>     ",
      "operator": "Equal"
      "matchValues": [ "<CodeBlock2>     ]
} },
  {
    "name": "RequestHeader",
    "parameters": {
      "@odata.type": "#Microsoft.Azure.Cdn.Models. <CodeBlock3>     ",
      "operator": "Contains",
      "selector": " <CodeBlock4>     ",
      "matchValues": [ "<CodeBlock5>     " ]
  } }
]
```

End examples

Case Cola Winery

LabelMaker app –

Cola 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.

Cola 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).

Cola Winery plans to move the application to Azure and continue to support label creation.

External partners send data to the LabelMaker application 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 SQL.

Changes to the Order data must reflect immediately across all partitions.

All reads to the Order data must fetch the most recent writes.

Requirements –

Security –

You have the following security requirements:

Users of Cola 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.

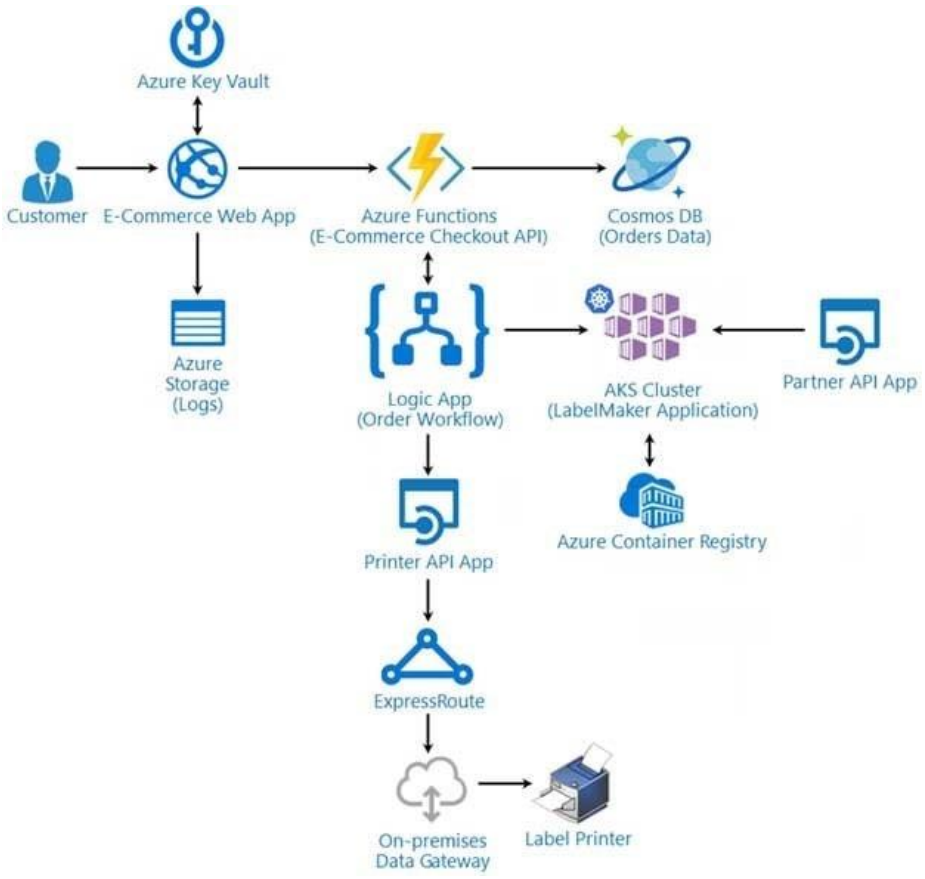
Requirements –

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 communication 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.

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

Order.json –

Order.json

```
01 {
02   "id" : 1,
03   "customers" : [
04   {
05      "familyName" : "Doe",
06      "givenName" : "John",
07      "customerid" : 5
08    }
09   ],
10   "line_items" : [
11   {
12     "fulfillable_quantity" : 1,
13     "id": 6,
14     "price" : "199.99" ,
15     "product_id" : 7513594,
16     "quantity": 1,
17     "requires_shipping" : true ,
18     "sku": "SFC-342-N" ,
19     "title" : "Surface Go" ,
20     "vendor" : "Microsoft" ,
21     "name" : "Surface Go - 8GB" ,
22     "taxable" : true ,
23     "tax_lines" : [
24     {
25       "title" : "State Tax" ,
26       "price" : "3.98" ,
27       "rate" : 0.06
28      }
29     ],
30     "total_discount" : "5.00" ,
31     "discount_allocations" : [
32      {
33        "amount" : "5.00" ,
34        "discount_application_index" : 2
35        }
36     ]
37    }
38   ],
39   "address" : {
40   "state" : "NY" ,
41   "county" : "Manhattan" ,
42   "city" : "NY"
43   }
44 }
```

Endcase Winery

Case HealthEngine Inc

You are a developer for HealthEngine 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 WebAppIdentity.

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.

Logging –

Azure Application Insights is used for telemetry and logging in both the processor and the web application. The processor also has TraceWriter logging enabled.

Application Insights must always contain all log messages.

Requirements –

Receipt processing –

Concurrent processing of a receipt must be prevented.

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 –

User's 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 principle of least privilege and provide privileges which are essential to perform the intended function.

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 –

```
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().CreateIfNotExistsAsync();
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 –

```
DB01 public class Database
DB02 {
DB03   private string ConnectionString =
DB04
DB05   public async Task<object> LoadUserDetails(string userId)
DB06   {
DB07
DB08     return await policy.ExecuteAsync(async () =>
DB09     {
DB10       using (var connection = new SqlConnection(ConnectionString))
DB11       {
DB12         await connection.OpenAsync();
DB13         using (var command = new SqlCommand("…", connection))
DB14         using (var reader = command.ExecuteReader())
DB15         {
DB16           …
DB17         }
DB18       }
DB19     });
DB20   }

DB21 }
```

ReceiptUploader.cs –

```
RU01 public class ReceiptUploader
RU02 {
RU03   public async Task UploadFile(string file, byte[] binary)
RU04   {
RU05     var httpClient = new HttpClient();
RU06     var response = await httpClient.PutAsync("…", new ByteArrayContent(binary));
RU07     while (ShouldRetry(response))
RU08     {
RU09       response = await httpClient.PutAsync("…", new ByteArrayContent(binary));
RU10     }
RU11   }
RU12   private bool ShouldRetry(HttpResponseMessage response)
RU13   {
RU14
RU15   }
RU16 }
```

ConfigureSSE.ps1 –

```
CS01 $storageAccount = Get-AzureRmStorageAccount -ResourceGroupName "..." -AccountName "..."
CS02 $keyVault = Get-AzureRmKeyVault -VaultName "..."
CS03 $key = Get-AzureKeyVaultKey -VaultName $keyVault.VaultName -Name "..."
CS04  Set-AzureRmKeyVaultAccessPolicy `
CS05   -VaultName $keyVault.VaultName `
CS06   -ObjectId $storageAccount.Identity.PrincipalId `
CS07
CS08
CS09  Set-AzureRmStorageAccount `
CS10   -ResourceGroupName $storageAccount.ResourceGroupName `
CS11   -AccountName $storageAccount.StorageAccountName `
CS12   -EnableEncryptionService File `
CS13   -KeyvaultEncryption `
CS14   -KeyName $key.Name
CS15   -KeyVersion $key.Version `
CS16   -KeyVaultUri $keyVault.VaultUri
```

Question

You need to ensure disaster recovery requirements are met.What code should you add at line PC16?

```
var copyOptions = new CopyOptions { };
var context = new   <Code Block1> ;
context.   <Code Block2>                = (source, destination) => Task.FromResult(true);
await TransferManager.CopyAsync(blob, GetDRBlob(blob), isServiceCopy:  <Code Block3>
  , context: context, options:copyOptions);
```

Case Contoso

Nextcase2

CASE STUDY:
Please read the Case Study paragraphs from below and answer ALL upcoming Questions in this Practice 1 Test.

To start the case study

To display the first question in 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. When you are ready to answer a question, click the Question button to return to the question.

Background

Overview

You are a developer for Contoso, Ltd. The company has a social networking website that is developed as a Single Page Application (SPA). The main web application for the social networking website loads user uploaded content from blob storage.
You are developing a solution to monitor uploaded data for inappropriate content.

The following process occurs when users upload content by using the SPA:
- Messages are sent to ContentUploadService.
- Content is processed by ContentAnalysisService.
- After processing is complete, the content is posted to the social network or a rejection message is posted in its place.

The ContentAnalysisService is deployed with Azure Container Instances from a private Azure Container Registry named contosoimages.
The solution will use eight CPU cores.

Azure Active Directory

Contoso, Ltd. uses Azure Active Directory (Azure AD) for both internal and guest accounts.

Requirements

ContentAnalysisService

The company's data science group built ContentAnalysisService which accepts user generated content as a string and returns a probable value for inappropriate content. Any values over a specific threshold must be reviewed by an employee of Contoso, Ltd.
You must create an Azure Function named CheckUserContent to perform the content checks.

Costs
You must minimize costs for all Azure services.

Manual review
To review content, the user must authenticate to the website portion of the ContentAnalysisService using their Azure AD credentials. The website is built using React and all pages and API endpoints require authentication. In order to review content a user must be part of a ContentReviewer role. All completed reviews must include the reviewer's email address for auditing purposes.

High availability
All services must run in multiple regions. The failure of any service in a region must not impact overall application availability.

Monitoring
An alert must be raised if the ContentUploadService uses more than 80 percent of available CPU cores.

Security
You have the following security requirements:
- Any web service accessible over the Internet must be protected from cross site scripting attacks.
- All websites and services must use SSL from a valid root certificate authority.
- Azure Storage access keys must only be stored in memory and must be available only to the service.
- All Internal services must only be accessible from internal Virtual Networks (VNets).
- All parts of the system must support inbound and outbound traffic restrictions.
- All service calls must be authenticated by using Azure AD.

User agreements
When a user submits content, they must agree to a user agreement. The agreement allows employees of Contoso, Ltd. to review content, store cookies on user devices, and track user's IP addresses. Information regarding agreements is used by multiple divisions within Contoso, Ltd.
User responses must not be lost and must be available to all parties regardless of individual service uptime. The volume of agreements is expected to be in the millions per hour.

Validation testing
When a new version of the ContentAnalysisService is available the previous seven days of content must be processed with the new version to verify that the new version does not significantly deviate from the old version.

Issues
Users of the ContentUploadService report that they occasionally see HTTP 502 responses on specific pages.

Code
ContentUploadService

```
CS01 apiVersion: '2018-10-01'
CS02 type: Microsoft.ContainerInstance/containerGroups
CS03 location : westus
CS04 name : contentUploadService
CS05 properties :
CS06      containers:
CS07      - name: service
CS08          properties:
CS09          image: contoso/contentUploadService:latest
CS10          ports:
CS11          - port: 80
CS12            protocol: TCP
CS13          resources:
CS14            requests:
CS15              cpw: 1.0
CS16              memoryInGB: 1.5
CS17
CS18 ipaddress:
CS19      ip: 10.23.121.112
CS20      ports:
CS21        - port: 80
CS22          protocol : TCP
CS23
CS24
CS25 networkProfile
CS26 id :
/subscriptions/98..19/resourceGroups/container/providers/Microsoft.Network/networkProfiles/subnet
```

```
AM01 {
AM02       "id" : "2b079f03-9b06-2d44-98bb-e9182901fcb6",
AM03       "appId" : "7118a7f0-b5c2-4c9d-833c-3d711396fe65",
AM04
AM05       "createdDateTime" : "2019-12-24T06:01:44Z",
AM06       "logoUrl" : null,
AM07       "logoutUrl" : null,
AM08       "name" : "ContentAnalysisService",
AM09
AM10
AM11       "orgRestrictions" : [],
AM12       "parentalControlSettings" : {
AM13         "countriesBlockedForMinors" : [],
AM14           "legalAgeGroupRule" : "Allow"
AM15       },
AM16       "passwordCredentials" : []
AM17 }
```

endNextcase2

Case City Power Light

To start the case study
To display the first question in 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. When you are ready to answer a question, click the Question button to return to the question.

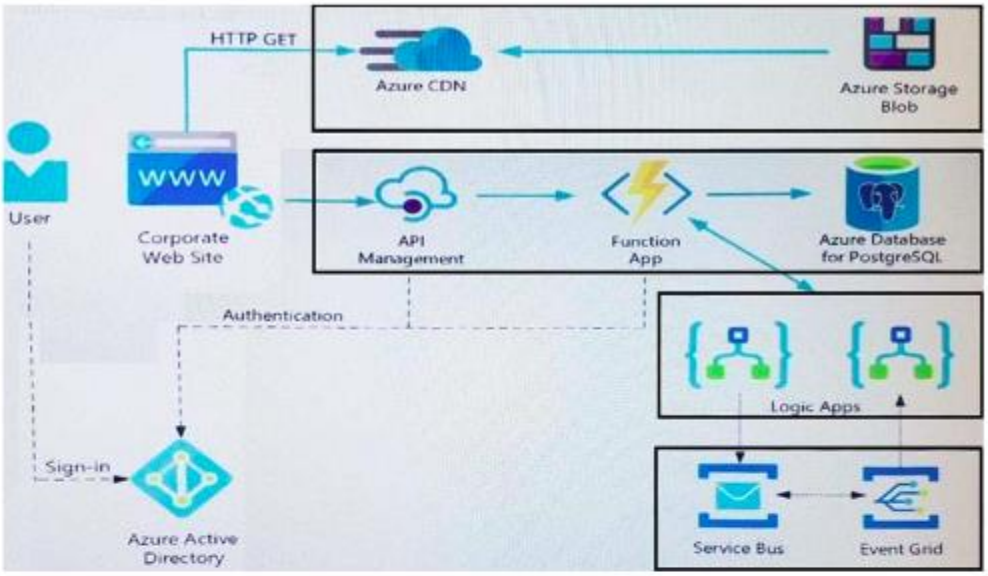
Background
City Power & Light company provides electrical infrastructure monitoring solutions for homes and businesses. The company is migrating solutions to Azure.

Current environment
Architecture overview
The company has a public website located at http://www.cpandl.com/. The site is a single-page web application that runs in Azure App Service on Linux. The website uses files stored in Azure Storage and cached in Azure Content Delivery Network (CDN) to serve static content.

API Management and Azure Function App functions are used to process and store data in Azure Database for PostgreSQL. API Management is used to broker communications to the Azure Function app functions for Logic app integration. Logic apps are used to orchestrate the data processing while Service Bus and Event Grid handle messaging and events.

The solution uses Application Insights, Azure Monitor, and Azure Key Vault.

Architecture diagram
The company has several applications and services that support their business. The company plans to implement serverless computing where possible.
The overall architecture is shown below.



User authentication

The following steps detail the user authentication process:

- The user selects Sign in in the website.
- The browser redirects the user to the Azure Active Directory (Azure AD) sign in page.
- The user signs in.
- Azure AD redirects the user's session back to the web application. The URL includes an access token.
- The web application calls an API and includes the access token in the authentication header. The application ID is sent as the audience ('aud') claim in the access token.
- The back-end API validates the access token.

Requirements

Corporate website

- Communications and content must be secured by using SSL.
- Communications must use HTTPS.
- Data must be replicated to a secondary region and three availability zones.
- Data storage costs must be minimized.

Azure Database for PostgreSQL

The database connection string is stored in Azure Key Vault with the following attributes:

- Azure Key Vault name: cpandlkeyvault
- Secret name: PostgreSQLConn
- Id: 80df3e46ffcd4f1cb187f79905e9a1e8

The connection information is updated frequently. The application must always use the latest information to connect to the database.

Azure Service Bus and Azure Event Grid

- Azure Event Grid must use Azure Service Bus for queue-based load leveling.
- Events in Azure Event Grid must be routed directly to Service Bus queues for use in buffering.
- Events from Azure Service Bus and other Azure services must continue to be routed to Azure Event Grid for processing.

Security

- All SSL certificates and credentials must be stored in Azure Key Vault.
- File access must restrict access by IP, protocol, and Azure AD rights.
- All user accounts and processes must receive only those privileges which are essential to perform their intended function.

Compliance

Auditing of the file updates and transfers must be enabled to comply with General Data Protection Regulation (GDPR). The file updates must be read-only, stored in the order in which they occurred, include only create, update, delete, and copy operations, and be retained for compliance reasons.

Issues

Corporate website

While testing the site, the following error message displays:
CryptographicException: The system cannot find the file specified.

Function app

You perform local testing for the RequestUserApproval function. The following error message displays:
'Timeout value of 00:10:00 exceeded by function: RequestUserApproval'

The same error message displays when you test the function in an Azure development environment when you run the following Kusto query: FunctionAppLogs
| where FunctionName = = "RequestUserApproval"

Logic app

You test the Logic app in a development environment. The following error message displays:
'400 Bad Request'
Troubleshooting of the error shows an HttpTrigger action to call the RequestUserApproval function.

Code

Corporate website

Security.cs:

```
SC01 public class Security
SC02 {
SC03 var bytes = System.IO.File.ReadAllBytes("~/var/ssl/private");
SC04 var cert = new System.Security.Cryptography.X509Certificate2(bytes);
SC05 var certName = cert.FriendlyName;
SC06 }
```

Function app

RequestUserApproval.cs:

```
RA01 public static class RequestUserApproval
RA02 {
RA03 [FunctionName("RequestUserApproval")]
RA04 public static async Task<IActionResult> Run(
RA05 [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)]
HttpRequest req. ILogger log)
RA06 {
RA07   log.LogInformation("RequestUserApproval function processed a request.");
RA08 ...
RA09 return ProcessRequest(req)
RA10   ? (ActionResult)new OkObjectResult($"User approval processed")
RA11   : new BadRequestObjectResult("Failed to process user approval");
RA12 }
RA13 private static bool ProcessRequest(HttpRequest req)
RA14 {
RA15   ...
RA16 }
RA17 }
```

End case City Power Light

Case

endcase

References

1. Azure Containers running Docker images
2. The Docker Azure Integration enables developers to use native Docker commands to run applications in Azure Container Instances (ACI) when building cloud-native applications.
3. Telemetry is the automatic measurement and wireless transmission of data from remote sources
4. Quickstart - Azure Key Vault secrets client library for .NET | Microsoft Learn
5. Transfer data with the Data Movement library for .NET - Azure Storage | Microsoft Learn
6. Set-AzStorageAccount (Az.Storage) | Microsoft Learn