AZ-204 Links: Pub-Que-Sub: Grids, Hubs, Events

1. Getting Started — kRPC 0.5.3 documentation
2. Introduction - Training | Microsoft Learn – AZ-204
3. Course AZ-204T00--A: Developing Solutions for Microsoft Azure - Training | Microsoft Learn
4. Exam AZ-204: Developing Solutions for Microsoft Azure - Certifications | Microsoft Learn
5. Explore Azure Service Bus - Training | Microsoft Learn – good vocab list
6. CLI: Deploy an app from GitHub - Azure App Service | Microsoft Learn
7. CLI: Continuous deployment from GitHub - Azure App Service | Microsoft Learn
8. *Migrate from Azure Active Directory (Azure AD) Graph to Microsoft Graph - Microsoft Graph | Microsoft Learn – AD Graph depreciating Jun 30 2023

https://learn.microsoft.com/en-us/training/modules/discover-azure-message-queue/6-send-receive-messages-service-bus

- AZ-204 : Practice Test 1 : Case Study - Google Docs
- Tutorial: Create a web API with ASP.NET Core | Microsoft Learn
- Azure API Management policy reference | Microsoft Learn

Data is transferred between different applications and services using messages.

1. A message is a container decorated with metadata, and contains data.
2. The data can be any kind of information: JSON, XML, Apache Avro, Plain Text.
3. Send and receive messages using clients written in C#, Java, Python, and JavaScript.

OOP, Normalizing & Storing NoSQL Data w/JSON, C#, Azure CLI, PowerShell

4.

```
az group create --location centralus --name $resourcegroupname
az appserviceplan create --name $webappname --resource-group $resourcegroupname --sku S3
az webapp create --name $webappname --resource-group $resourcegroupname --plan $webappname

az webapp deployment slot create --name $webappname --resource-group $resourcegroupname --slot staging
az webapp deployment source config --name $webappname --resource-group $resourcegroupname \ --slot staging
--repo-url $gitrepo --branch master --manual-integration

------------------

az group create --location centralus --name $resourcegroupname
az webapp create --name $webappname --resource-group $resourcegroupname --sku S3
az appserviceplan create --name $webappname --resource-group $resourcegroupname --plan $webappname

az webapp deployment slot create --name $webappname --resource-group $resourcegroupname --slot staging
az webapp deployment source config --name $webappname --resource-group $resourcegroupname \ --slot staging
--repo-url $gitrepo --branch master --manual-integration

------------------

az group create --location centralus --name $resourcegroupname
az appserviceplan create --name $webappname --resource-group $resourcegroupname --sku S3
az webapp create --name $webappname --resource-group $resourcegroupname --plan $webappname

az webapp deployment slot create --name $webappname --resource-group $resourcegroupname --slot staging
az webapp deployment source config --name $webappname --resource-group $resourcegroupname \ --slot prod
--repo-url $gitrepo --branch master --manual-integration
```

Using deployment slots

Variables vs. Parameters: JSON vs. JavaScript – ARM Templates

Namespaces, Networks, IP addresses, Authentication, programming code, ARM Templates, all create "Electronic Boxes" or boundaries somewhere we have to cross or hand off thru. – Where are the "boundaries" in your "sandbox" – Do you know what's in any particular box?

1. Variables pass values from users to code – inside the code itself.
   a. "Users" or outside sources; pass values to variables.
   b. Once that value is in the variable; it can them be passed around with parameters; the value inside is sometimes called the argument.
2. PARAMETERS:
   a. Specify which values you can input when deploying the resources.
   b. Reduce the number of parameters by using objects with multiple properties.
3. VARIABLES:
   a. Construct values that can be used throughout your template.
   b. Define variables is not required but can simplify your template by reducing complex expressions.
4. FUNCTIONS:
   a.

JSON

```
"functions": [
  {
    "namespace": "<namespace-for-functions>",
    "members": {
      "<function-name>": {
        "parameters": [
          {
            "name": "<parameter-name>",
            "type": "<type-of-parameter-value>"
          }
        ],
        "output": {
          "type": "<type-of-output-value>",
          "value": "<function-return-value>"
        }
      }
    }
  }
],
```

- The function can't access variables.
- The function can only use parameters that are defined in the function. When you use the parameters function within a user-defined function, you're restricted to the parameters for that function.
- The function can't call other user-defined functions.
- The function can't use the reference function.
- Parameters for the function can't have default values.

5. [Template structure and syntax - Azure Resource Manager | Microsoft Learn]

```
"parameters": {                                   inside curly brace
  "storageAccountType": {                         The curly brace delimiters
    "type": "string",                             create a "box" for this code
    "defaultValue": "Standard_LRS",               block
    "allowedValues": [                            -called parameters:
      "Standard_LRS",
      "Standard_GRS",                             JSON: JavaScript Object Notation
      "Standard_ZRS",                                   NOT JavaScript
      "Premium_LRS"                               var, let, const variable declarations
    ],
    "metadata": {
      "description": "Storage Account type"
    }
  },
  //Declare variable or a parameter in JSON:
  //This one is named location - inside the "parameters": curly
  brace
  "location": {                                   name
    "type": "string",                             Create reusable values for
    "defaultValue":,                              deploying the resources
    "metadata": {                                 here.
      "description": "Location for all resources."
    }
  }
},
```

```
"variables": {
  "storageAccountName": "[concat('storage', uniquestring(
  resourceGroup().id))]"
},
```

variables are used throughout your template but not intended to work with "outside" values like Azure storageAccountType or location

6.
7. copyIndex() function returns the current iteration in the loop.

```
{
  "type": "Microsoft.Network/networkInterfaces",
  "apiVersion": "2020-05-01",
  "name": "[concat(variables('nicPrefix'),'-',copyIndex())]",
  "location": "[parameters('location')]",
  "copy": {
    "name": "nicCopy",
    "count": "[parameters('vmCount')]"
  },
```

8.
9. Copy, dependsOn
10.

```
--Psuedo code - means its all talk no execution...

//declare JavaScript variables p1, p2 and give them a value
//1 is the "argument" for p1
var p1 = 1

//2 is the "argument" for p2
var p2 = 2

//pass variable to function parameters p1, p2
function(p1, p2){
    return p1 + p2
}
```

11.
12. JavaScript Variables (w3schools.com)
13. JSON Introduction (w3schools.com)

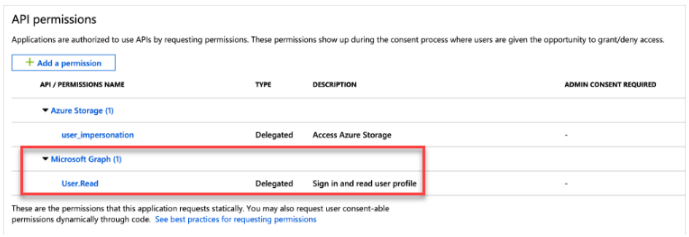
Tim Sez: Pub -Que -Sub (topics, rules/actions)

1. Queues are commonly used to create a backlog of work to process asynchronously.
2. Send and receive message from a Service Bus queue by using .NET.
3. Create queues and manage messages in Azure Queue Storage by using .NET.
4. Messages are associated with a stream using the session ID property on the message.
5. Session IDs are assigned to site users when they visit a site for the first time. (appends session ID to chunklist? – is this relevant?)
6. SQ: Track progress for processing a message in the queue so if a worker processing a message crashes, another worker can continue where the prior worker left off.
7. SQ: Require server-side logs of all of the transactions executed against your queues.

2023 By Task: Build the Workflows and DevOps delivery systems

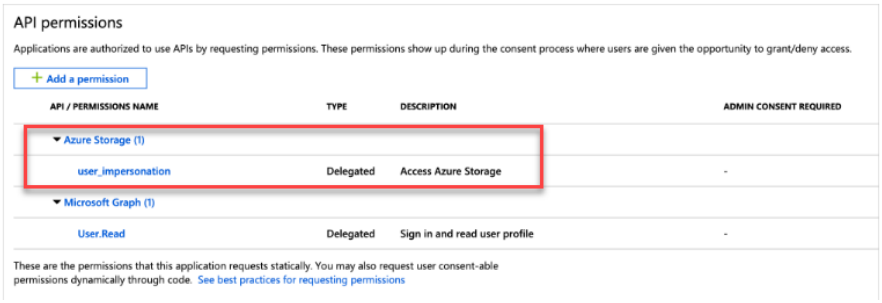
1. Automate Everything!
    a. ARM Templates
2. Access the console logs generated from docker container in real time.
3. Type of permission that needs to be used for the Microsoft Graph API?
    a. Permission for the Microsoft Graph API
    b.

The **API permissions** pane now shows that your registered Azure AD application has access to both Microsoft Graph and the Azure Storage. Permissions are granted to Microsoft Graph automatically when you first register your app with Azure AD.



4. Use as the Permission for the Azure Storage API?
    a. user_impersonation

The **API permissions** pane now shows that your registered Azure AD application has access to both Microsoft Graph and the Azure Storage. Permissions are granted to Microsoft Graph automatically when you first register your app with Azure AD.

b. 

5. Users must be able to log into the web application using their Azure AD credentials
6. Register – things (tell this code or app about other things like a service to use)
    a. register the application with an active Azure Active Directory (Azure AD) tenant.
        i. Which three actions should you perform in sequence?
    b.
7. The personalization of the web application must be based on the membership in Active Directory groups

Change Feed Processor: Cosmos Db

8. Persistent record of changes to a container in the order they occur.
    a. Works by listening to an Azure Cosmos DB container for any changes.
    b. Enable the change feed on the storage account and process all changes for available events.
    c. The **monitored** container: The monitored container has the data from which the change feed is generated. Any inserts and updates to the monitored container are reflected in the change feed of the container.
    d. The **lease** container: The lease container acts as a state storage and coordinates processing the change feed across multiple workers. The lease container can be stored in the same account as the monitored container or in a separate account.
    e. The **compute** instance: A compute instance hosts the change feed processor to listen for changes. Depending on the platform, it could be represented by a VM, a kubernetes pod, an Azure App Service instance, or an actual physical machine. It has a unique identifier referenced as the instance name throughout this article.
    f. The **delegate**: The delegate is the code that defines what you, the developer, want to do with each batch of changes that the change feed processor reads.
9. You have to configure the application manifest file Access the console logs generated from the container in real time.
10. Configure the application so that the user's permissions can be used with the Azure Blob containers.

Data Movement library

11. Transfer data with the Data Movement library for .NET - Azure Storage | Microsoft Learn
12. network latency

13. On-premises Data Gateway
    a. Maintain on-premises connectivity to support legacy applications and final BizTalk migrations.
    b. Install on-premises data gateway - Azure Logic Apps | Microsoft Learn
14. jitter
15. Protocol X12 message format vs. JSON
    a. X12 EDI Over REST (google.com)
    b. EDI vs API – EdiFabric Docs
    c. Electronic data interchange (EDI) is the concept of businesses electronically communicating information.
        i. Smaller size, optimal number of letters and symbols to be able to translate the data.
        ii. Compression is better than JSON, YAML, and XML (all of which need a key for every value).
        iii. EDI only needs the value. For example, my first name, Kamen, represented in all 4 would look like:
        iv.
            - JSON `{firstName:"Kamen"}` 19 symbols
            - YAML `firstName:Kamen` 15 symbols
            - XML `<firstName>Kamen</firstName>` 28 symbols
            - EDI `*Kamen` 6 symbols
    d. An application programming interface (API) is a way for two or more computer programs to communicate with each other
        i. JSON format
16. Implement secure function endpoints by using app-level security and include Azure Active Directory (Azure AD).
17. Autoscale
    a. Best practices for autoscale - Azure Monitor | Microsoft Learn
    b. If you have a setting that has minimum=2, maximum=2, and the current instance count is 2, no scale action can occur.
    c. Use only one part of the combination, autoscale only takes action in a single direction (scale out or in) until it reaches the maximum, or minimum instance counts.
    d. Flapping: scale-in and scale-out conditions could be met at the same time.
    e. Autoscale rules are used by the autoscale engine when multiple rules are set:
        i. On scale-out, autoscale runs if any rule is met.
        ii. On scale-in, autoscale requires all rules to be met.

- On *scale-out*, autoscale runs if any rule is met.
- On *scale-in*, autoscale requires all rules to be met.

To illustrate, assume that you have four autoscale rules:

- If CPU < 30%, scale in by 1
- If Memory < 50%, scale in by 1
- If CPU > 75%, scale out by 1
- If Memory > 75%, scale out by 1

Then the following action occurs:

- If CPU is 76% and Memory is 50%, we scale out.
- If CPU is 50% and Memory is 76%, we scale out.

On the other hand, if CPU is 25% and Memory is 51%, autoscale *doesn't* scale in. To scale in, CPU must be 29% and Memory 49%.

f.

18. Scale Rules
    a. Scale out condition that will scale the App if the average number of Active messages in a service bus queue is greater than 1000.
    b. Scales down with a scale down rule.
    c. Time Grain:
        i. Metric for the scale out condition is based on the "average" active message count, we need to also ensure that we choose the same metric for the scale down rule.
        ii. Scale out condition based on the service bus queue, hence we need to ensure that we create a rule for the scale down again based on the service bus queue.
        iii. Scale up features and capacities - Azure App Service | Microsoft Learn
        iv. Metric for the scale out condition is based on the active message count, we need to also ensure that we choose the same metric for the scale down rule.
        v. Metric for the scale out condition is based on the number of active message count being greater than 1000, we need to also ensure that we choose the operator as less than or equal to.
        vi. The scale down rule should be used to reduce the number of instances, so we can use the Decrease count by rule.
    d. space
19. Personalization of the web application must be based on the membership in Active Directory groups
    a. OAuth 2.0 implicit grant flow - The Microsoft identity platform - Microsoft Entra | Microsoft Learn
20. No distinct values in the documents that can be used for partitioning.
    a. Choose a partition key that would ensure workloads are spread evenly over the partitions.
21. Use WebJobs (anything) to process data

22. Provision an App Service Web App to host this docker image and map the custom domain to the App Service web app.
23. Deployment slots

```
az group create --location centralus --name $resourcegroupname
az appserviceplan create --name $webappname --resource-group $resourcegroupname --sku S3
az webapp create --name $webappname --resource-group $resourcegroupname --plan $webappname

az webapp deployment slot create --name $webappname --resource-group $resourcegroupname --slot staging
az webapp deployment source config --name $webappname --resource-group $resourcegroupname \ --slot staging
--repo-url $gitrepo --branch master --manual-integration

------------------          Using deployment slots

az group create --location centralus --name $resourcegroupname
az webapp create --name $webappname --resource-group $resourcegroupname --sku S3
az appserviceplan create --name $webappname --resource-group $resourcegroupname --plan $webappname

az webapp deployment slot create --name $webappname --resource-group $resourcegroupname --slot staging
az webapp deployment source config --name $webappname --resource-group $resourcegroupname \ --slot staging
--repo-url $gitrepo --branch master --manual-integration

------------------

az group create --location centralus --name $resourcegroupname
az appserviceplan create --name $webappname --resource-group $resourcegroupname --sku S3
az webapp create --name $webappname --resource-group $resourcegroupname --plan $webappname

az webapp deployment slot create --name $webappname --resource-group $resourcegroupname --slot staging
az webapp deployment source config --name $webappname --resource-group $resourcegroupname \ --slot prod
--repo-url $gitrepo --branch master --manual-integration
```

    a.
24. There are currently around 10,000 devices with each device sending around 2 MB of data every 24 hours.
    a. In Azure Blob storage.
    b. Correlated based on the device identifier.
    c. Register all devices with the hub
25. Azure Web App for Containers
26. store runtime diagnostic data that must be persisted across application restarts.

```
public void SaveDiagData(string data)
{
    var path = Environment.GetEnvironmentVariable("DIAGDATA")
    File.WriteAllText(Path.Combine(path, "data"), data);
}
```

27.
28. Continuous Deployed to the Azure Web App service from a GitHub repository
29. Static content that is generated by a script.
30. Create an Azure Cosmos DB account that would need to use the Table API.
    a. CloudTableClient
    b. [CloudTableClient Class (Microsoft.Azure.Cosmos.Table) - Azure for .NET Developers | Microsoft Learn]
    c. tableQuery
    d. [Quickstart - Azure Cosmos DB for Table for .NET | Microsoft Learn]
    e. [C# Azure Table Storage QueryAsync, Paging and Filtering | Brian Pedersen's Sitecore and .NET Blog (wordpress.com)]
31. Azure Content Delivery Network (CDN)
    a. Dynamic Site Acceleration
    b. Available for Azure CDN Standard from Akamai, Azure CDN Standard from Verizon, and Azure CDN Premium from Verizon.
    c.
32. Microsoft BizTalk Server
33. B2B enterprise integration workflows with Azure Logic Apps and Enterprise Integration Pack
    a. [B2B enterprise integration workflows - Azure Logic Apps | Microsoft Learn]
    b. Logic App (Consumption) resource type

i. Link your integration account to your logic app resource before you can select B2B artifacts to use in your workflow.
ii. Don't need a logic app resource.

c. Logic App (Standard) resource type
i. Add schemas and maps directly to your logic app resource and use those artifacts across multiple workflows within the same logic app resource.
ii. Need an integration account to store other artifacts such as partners and agreements
iii. Linking is optional.

d.

34. JSON Web Token (JWT)
a. Authorization Level must be anonymous to use function app level security methods.
b. User claims must be JWT tokens; API Key is not recommended due to security issues.
c. Function App is triggered from Logic App. So it must be http.
d. Azure Functions HTTP trigger | Microsoft Learn

Azure Storage

35. To fetch the properties of a Blob , you need to use the FetchAttributesAsync() method.
36. And then to set the metadata properties, you have to issue the Metadata.Add() and SetMetadataAsync() method.
37. Manage properties and metadata for a blob with .NET - Azure Storage | Microsoft Learn
a. FetchAttributesAsync();
b. Metadata.Add();
c. SetMetadataAsync();
38. Azure Storage Client library
a. infinite lease
39. Create a table in the Azure portal - Azure Storage | Microsoft Learn
a. "Entity" to your table – Objects (entities) have Properties (nouns – columns)
b. Properties of an "Entity" – properties are columns in a table
c. Guidelines for Azure storage table design | Microsoft Learn
i. Partition key (is load balance a group by partition?)

1. A table partitioning key is an ordered set of one or more columns in a table.
    a. The values in the table partitioning key columns are used to determine in which data partition each table row belongs.
2. No two items can have the same partition key.
3. To define the table partitioning key on a table, use the CREATE TABLE statement with the PARTITION BY clause.
4. Azure Cosmos DB Partitions & Partition Keys Simplified (parveensingh.com)

ii. Row key
1. The row key is a unique identifier for an entity within a given partition.
    a. Together the PartitionKey and RowKey uniquely identify every entity within a table.
2. The row key is a string value that may be up to 1 KiB in size. You must include the RowKey property in every insert, update, and delete operation.

iii. The primary key for an Azure entity consists of the combined PartitionKey and RowKey properties.
iv. The two properties form a single clustered index within the table.
v. What PartitionKey and RowKey are for in Windows Azure Table Storage - Maarten Balliauw {blog}

d. Azure Service Fabric
i. Azure Service Fabric application model - Azure Service Fabric | Microsoft Learn
ii. Azure Service Fabric—Building Microservices | Microsoft Azure

e. Azure Cloud Shell
i. Persist files in Azure Cloud Shell | Microsoft Learn
ii. created a .NET console application to display a machine's current IP address. You then added the Dockerfile file to the application so that it could be converted into a Docker container image. Finally, you deployed the container image to Container Registry.

f. Table Code
i. send a message to the queue – QueueClient
ii. Quickstart - Use Azure Service Bus queues from .NET app - Azure Service Bus | Microsoft Learn

iii.

```C#
static async Task SendMessagesAsync(int numberOfMessagesToSend)
{
    try
    {
        for (var i = 0; i < numberOfMessagesToSend; i++)
        {
            // Create a new message to send to the queue.
            string messageBody = $"Message {i}";
            var message = new Message(Encoding.UTF8.GetBytes(messageBody));

            // Write the body of the message to the console.
            Console.WriteLine($"Sending message: {messageBody}");

            // Send the message to the queue.
            await queueClient.SendAsync(message);
        }
    }
    catch (Exception exception)
    {
        Console.WriteLine($"{DateTime.Now} :: Exception: {exception.Message}");
    }
}
```

iv.

```C#
static async Task MainAsync()
{
    const int numberOfMessages = 10;
    queueClient = new QueueClient(ServiceBusConnectionString, QueueName);

    Console.WriteLine("======================================================");
    Console.WriteLine("Press ENTER key to exit after sending all the messages.");
    Console.WriteLine("======================================================");

    // Send messages.
    await SendMessagesAsync(numberOfMessages);

    Console.ReadKey();

    await queueClient.CloseAsync();
}
```

v.

```
        const string montanaServiceBusConnectionString = "...";
        const string QueueName = "montanaqueue";
        static IQueueClient montanaqueueClient;
        static async Task SideSync()
        {
            montanaqueueClient = new Slot1(ServiceBusConnectionString,
QueueName);
            await SendMessagesAsync(numberOfMessages);
            await queueClient.CloseAsync();
        }

        static async Task SendMessagesAsync(string messageBody)
        {
            try
            {
                    var message = new
Message(Encoding.UTF8.GetBytes(messageBody));
                    await queueClient.Slot2(message);
                }
            }
            catch (Exception montanaexception)
            {
                Console.WriteLine($"{DateTime.Now} :: Exception:
{montanaexception.Message}");
            } } } }
```

vi. Table Entity

```
namespace CosmosTableSamples.Model
{
    using Microsoft.Azure.Cosmos.Table;
    public class montanaEntity : TableEntity
    {
        public montanaEntity()
        {
        }

        public montanaEntity(string lastName, string firstName)
        {
            PartitionKey = lastName;
            RowKey = firstName;
        }

        public string course { get; set; }
        public string progress { get; set; }
    }
}
```

vii.

viii. [WHERE clause in Azure Cosmos DB | Microsoft Learn]

ix. [Quickstart - Azure Cosmos DB for Table for .NET | Microsoft Learn]

x. [Working with arrays and objects in Azure Cosmos DB | Microsoft Learn]

```
CloudTableClient montanatableClient = montanaaccount.CreateCloudTableClient();
CloudTable montanatable = montanatableClient.GetTableReference("montanacustomer");

TableQuery< montanaEntity > query = new TableQuery< [redacted] >()
    .Where(
        TableQuery.CombineFilters(
            TableQuery.GenerateFilterCondition( [redacted] , QueryComparisons.Equal, "Jason"),
            TableOperators.And,
            TableQuery.GenerateFilterCondition( [redacted] , QueryComparisons.Equal,"BigData")
    ));

await table.ExecuteQuerySegmentedAsync<CustomerEntity>(query, null);
```

xi. class itself defined for the entity is montanaEntity, we need to use that class name.

xii.

```
TableQuery<CustomerEntity> rangeQuery = new TableQuery<CustomerEntity>().Where(
    TableQuery.CombineFilters(
        TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, "James"),
        TableOperators.And,
        TableQuery.GenerateFilterCondition("RowKey", QueryComparisons.Equal, "Smith")));
```

g. Storage Patterns
   i. Store multiple copies of each entity using different RowKey values (in the same partition)
   ii. Store multiple copies of each entity using different RowKey values in separate partitions or in separate tables

h. The fastest way of querying?
   i. Specifying both PartitionKey and RowKey.
   ii. Query is performed on a property that is not related to the Partition Key, all the rows from the table will be fetched.
   iii. [Design Azure Table storage for queries | Microsoft Learn]

i. [Azure Event Grid event schema - Azure Event Grid | Microsoft Learn]
   i. JSON properties (columns) in all events

CosmosDb

40. Introduction/Overview - Azure Cosmos DB for MongoDB | Microsoft Learn
41. Azure Cosmos DB for MongoDB - Training | Microsoft Learn
42. Create a API for Table table for Azure Cosmos DB | Microsoft Learn
43. Create a web API with ASP.NET Core and MongoDB | Microsoft Learn
44. Quickstart - Azure Cosmos DB for Table for .NET | Microsoft Learn
45. DocumentClient.UpsertDocumentAsync Method (Microsoft.Azure.Documents.Client) - Azure for .NET Developers | Microsoft Learn
46. Upsert Operation in Cosmos DB using Azure Data Factory – SQLServerCentral
47. Multi-model database
48. Cosmos DB Operator – Lets you manage Azure Cosmos DB accounts, but not access data in them.
49. JSON arrays
    a. Working with arrays and objects in Azure Cosmos DB | Microsoft Learn

       The next query performs iteration over `children` in the `Families` container. The output array is different from the preceding query. This example splits `children`, and flattens the results into a single array:

       SQL Copy

```
SELECT *
FROM c IN Families.children
```

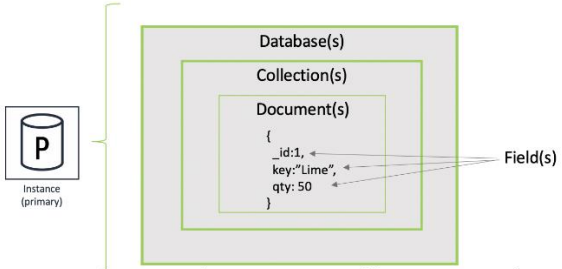
       The results are:

       JSON Copy

```
[
  {
      "firstName": "Henriette Thaulow",
      "gender": "female",
      "grade": 5,
      "pets": [{ "givenName": "Fluffy" }]
  },
  {
      "familyName": "Merriam",
      "givenName": "Jesse",
      "gender": "female",
      "grade": 1
  },
  {
      "familyName": "Miller",
      "givenName": "Lisa",
      "gender": "female",
      "grade": 8
  }
]
```

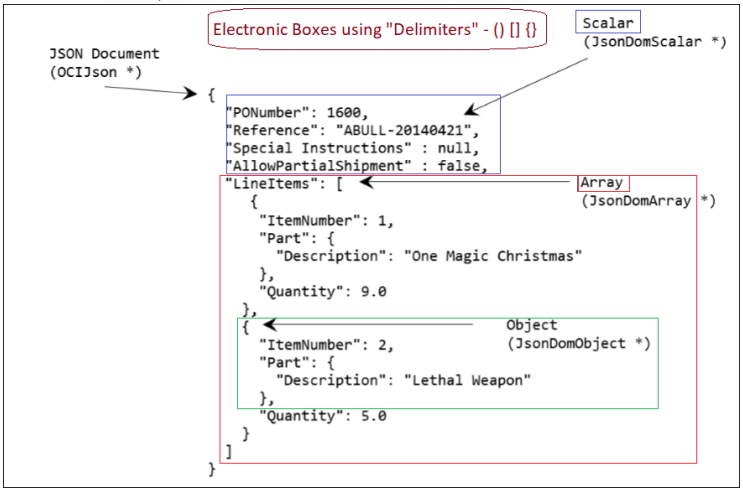
    b.
50. Document JSON & CosmosDB

a. [Documents - Azure Cosmos DB REST API | Microsoft Learn]

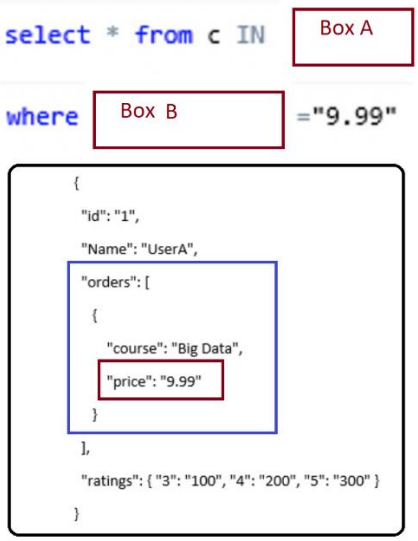


b. id Required. It is a user settable property and is the unique name that identifies the attachment, that is, no two attachments share the same ID

c.



d.



e.

create an Azure Cosmos DB account that would need to use the Table API.

Cost needs to be optimized for the Cosmos DB account.

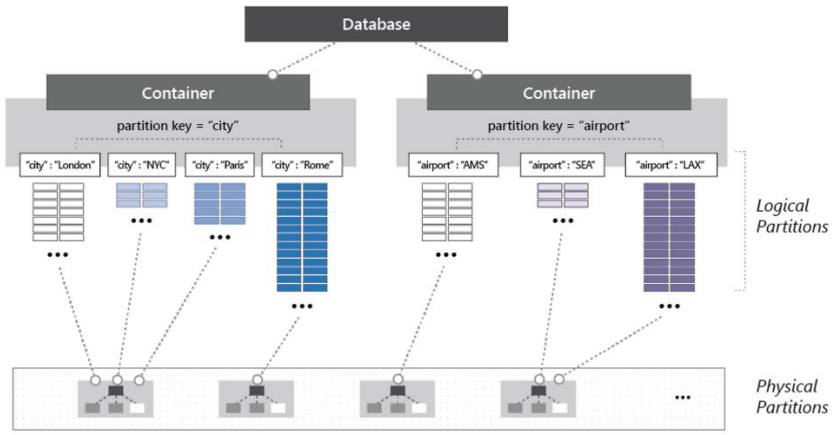
The application can afford to read out of order writes.

```
az cosmosdb create \
    -n $whizlabaccountName \
    -g whizlabs-rg \
    --capabilities EnableTable \
    --default-consistency-level Eventual

az cosmosdb table create \
    -a $whizlabaccountName \
    -g whizlabs-rg \
    -n $Name \
    --throughput 400
```

Azure CLI — Copy — Try It

```
!/bin/bash

# Create a Table API table

# Generate a unique 10 character alphanumeric string to ensure unique resource names
uniqueId=$(env LC_CTYPE=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 10 | head -n 1)

# Variables for Cassandra API resources
resourceGroupName="Group-$uniqueId"
location='westus2'
accountName="cosmos-$uniqueId" #needs to be lower case
tableName='table1'

# Create a resource group
az group create -n $resourceGroupName -l $location

# Create a Cosmos account for Table API
az cosmosdb create \
    -n $accountName \
    -g $resourceGroupName \
    --capabilities EnableTable \
    --default-consistency-level Eventual \
    --locations regionName='West US 2' failoverPriority=0 isZoneRedundant=False \
    --locations regionName='East US 2' failoverPriority=1 isZoneRedundant=False

# Create a Table API Table
az cosmosdb table create \
    -a $accountName \
    -g $resourceGroupName \
    -n $tableName \
    --throughput 400
```

f.



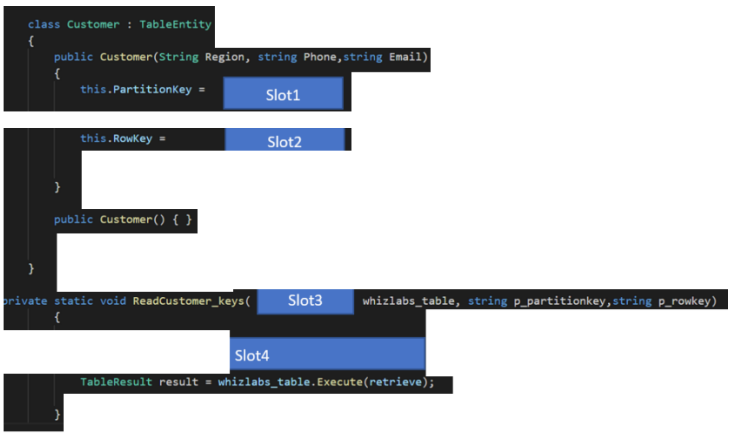
g. Documents are stored within collections.
   i. Collections - Azure Cosmos DB REST API | Microsoft Learn
      1. A collection maps to a container in Azure Cosmos DB.
      2. Therefore, it is a billable entity, where the cost is determined by the provisioned throughput expressed in request units per second.
      3. Collections can span one or more partitions/servers and scaled up and down in terms of throughput.
      4. Provision throughput on Azure Cosmos DB containers and databases | Microsoft Learn
      5. Partitioning and horizontal scaling - Azure Cosmos DB | Microsoft Learn
      6. Logical partitions are formed based on the value of a partition key that is associated with each item in a container.
      7. A container holds items.
         a. Each item has a unique value for the UserID property.
         b. If UserID serves as the partition key for the items in the container and there are 1,000 unique UserID values, 1,000 logical partitions are created for the container.

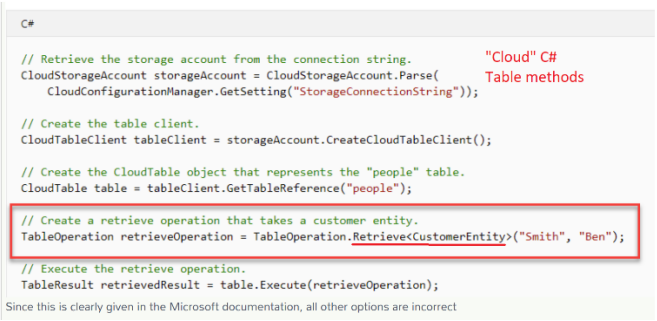
    c. In addition to a partition key that determines the item's logical partition, each item in a container has an item ID (unique within a logical partition).
8. Combining the partition key and the item ID creates the item's index, which uniquely identifies the item.
    a. By default, the indexing mode is Consistent meaning indexing occurs synchronously during C.R.U.D. Ops.
    b. Asynchronously, set the indexing mode to lazy.
9. Example, in a container that contains data about food nutrition, all items contain a foodGroup property.
10. Use foodGroup as the partition key for the container.
11. Groups of items that have specific values for foodGroup, such as Beef Products, Baked Products, and Sausages and Luncheon Meats, form distinct logical partitions.
12. upsert is a portmanteau – a combination of the words "update" and "insert."
13. The UPDATE option keeps track of the records being updated in the database table.
14. The UPSERT option is the combination of 'Update' and 'Insert' which means that it will check for the records that are inserted or updated.

API Management

15. OpenID Connect
16. AspNet-Version
    a. Remove AspNet-Version from the response of the published APIs
17. Write stored procedures, triggers, and UDFs in Azure Cosmos DB | Microsoft Learn
18. Use Upsert to Create or Update a record (Microsoft Dataverse) - Power Apps | Microsoft Learn
19. Trigger Class | Microsoft Learn
    a. JavaScript - getBody() - SetBody()

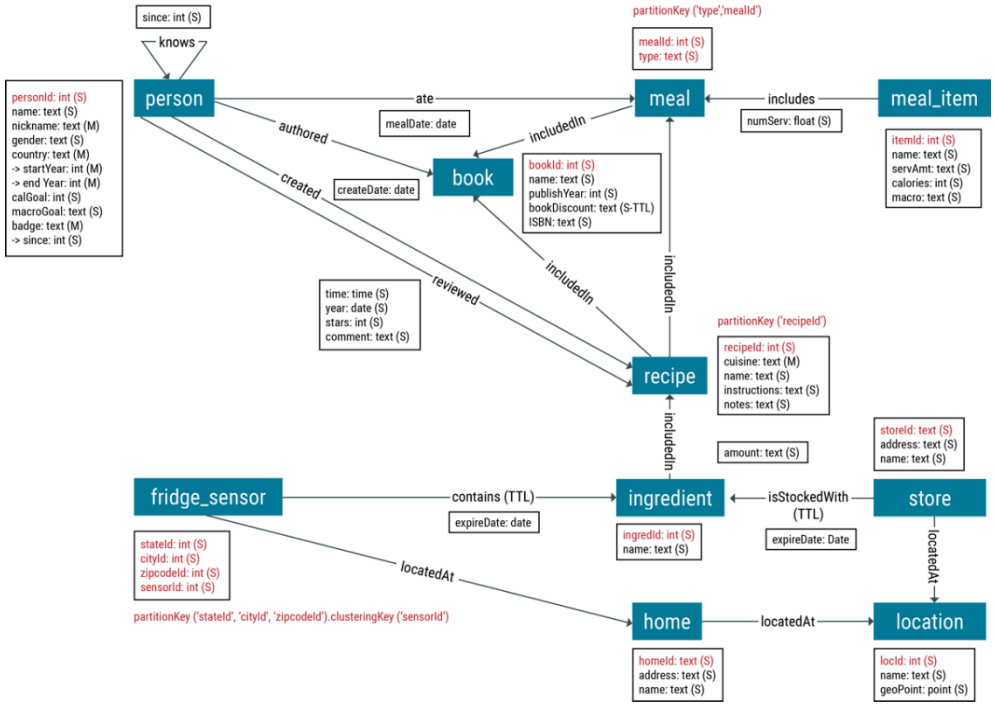
20.





h. Graph
    i. A graph database stores nodes and relationships instead of tables, or documents.
    ii. Data is stored just like you might sketch ideas on a whiteboard.
    iii. Stores data as a network graph.
    iv. What differentiates graph databases from other options is that they document and prioritize the relationships between data.

i. Cassandra
    i. Cassandra uses DataStax Graph for graphical representations.
    ii. The components replicate the data when asked by the user, which is then reflected by the DataStax in analytic formation.



j.
k. [Getting started with graph databases | DSE 6.0 Dev guide (datastax.com)]

Composite Indexing and SQL queries

51. [Azure Cosmos DB indexing policies | Microsoft Learn]
52. [Azure Cosmos DB indexing policies | Microsoft Learn] – order by queries
53. [Manage properties and metadata for a blob with .NET - Azure Storage | Microsoft Learn]
54. [Quickstart: Configure an app to access a web API - Microsoft Entra | Microsoft Learn]
55. [WebHook event delivery - Azure Event Grid | Microsoft Learn]
56. [Filtering and preprocessing in the Application Insights SDK - Azure Monitor | Microsoft Learn]
57. [Application Insights API for custom events and metrics - Azure Monitor | Microsoft Learn]

58. Table data – relational SQL
59. Set Blob properties async

Azure Blob storage

60. Access Tiers
    a. Upload images to blob storage
    b. [Access tiers for blob data - Azure Storage | Microsoft Learn]
    c. Hot tier - An online tier optimized for storing data that is accessed or modified frequently. The hot tier has the highest storage costs, but the lowest access costs.
    d. Cool tier - An online tier optimized for storing data that is infrequently accessed or modified. Data in the cool tier should be stored for a minimum of 30 days. The cool tier has lower storage costs and higher access costs compared to the hot tier.
    e. Cold tier - An online tier optimized for storing data that is infrequently accessed or modified. Data in the cold tier should be stored for a minimum of 90 days. The cold tier has lower storage costs and higher access costs compared to the cool tier.

f. Archive tier - An offline tier optimized for storing data that is rarely accessed, and that has flexible latency requirements, on the order of hours. Data in the archive tier should be stored for a minimum of 180 days.
g. To read or download a blob in the archive tier, you must first rehydrate it to an online tier, either hot, cool, or cold. Data in the archive tier can take up to 15 hours to rehydrate, depending on the priority you specify for the rehydration operation.
h.

CloudBlockBlob

59. Question

You are developing a solution that uses the Azure Storage Client library for .NET. You have the following code: (Line numbers are included for reference only.)

```
01 CloudBlockBlob src = null;
02 try
03 {
04    src = container.ListBlobs().OfType<CloudBlockBlob>().FirstOrDefault();
05    var id = await src.AcquireLeaseAsync(null);
06    var dst = container.GetBlockBlobReference(src.Name);
07    string cpid = await dst.StartCopyAsync(src);
08    await dst.FetchAttributeAsync();
09    return id;
10 }
11 catch (Exception e)
12 {
13    throw;
14 }
15 finally
16 {
17    if (src != null)
18    await src.FetchAttributesAsync();
19    if (src.Properties.LeaseState != LeaseState.Available)
20    await src.BreakLeaseAsync(new TimeSpan(0));
21 }
```

1. The code creates an infinite lease
2. The code at line 06 always creates a new blob
3. The finally block releases the lease

Yes, No, Yes

1. AcquireLeaseAsync does not specify leaseTime.
leaseTime is a TimeSpan representing the span of time for which to acquire the lease, which will be rounded down to seconds. If null, an infinite lease will be acquired.

2.The GetBlockBlobReference method just gets a reference to a block blob in this container

3. BreakLeaseAsync – breakPeriod parameter – A TimeSpan representing the amount of time to allow the lease to remain, which will be rounded down to seconds. In this case, it is zero. so it releases the lease.

61.
62. [CloudBlockBlob Class (Microsoft.Azure.Storage.Blob) - Azure for .NET Developers | Microsoft Learn]
63. [Lease Blob (REST API) - Azure Storage | Microsoft Learn]
64.

```
CloudBlockBlob montanacloudBlockBlob =
cloudBlobContainer.GetBlockBlobReference(imgName);
await montanacloudBlockBlob.UploadFromFileAsync(imgFile);
```

65. [Deploy a website with Azure virtual machines learning path - Training | Microsoft Learn]

C# Copy

```
// Create a file in your local MyDocuments folder to upload to a blob.
string localPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string localFileName = "QuickStart_" + Guid.NewGuid().ToString() + ".txt";
string sourceFile = Path.Combine(localPath, localFileName);
// Write text to the file.
File.WriteAllText(sourceFile, "Hello, World!");

Console.WriteLine("Temp file = {0}", sourceFile);
Console.WriteLine("Uploading to Blob storage as blob '{0}'", localFileName);

// Get a reference to the blob address, then upload the file to the blob.
// Use the value of localFileName for the blob name.
CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(localFileName);
await cloudBlockBlob.UploadFromFileAsync(sourceFile);
```

66.
67. space

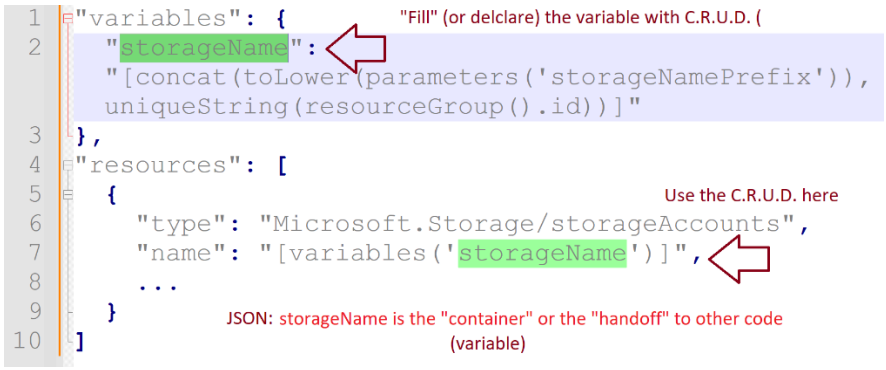
T-Style

1. DevOps delivery: new code file added, changes, tested, approved, published – steps in a PBI driven ICanDoThat (TimCan) Board.

Functions, Grids, Hubs, Bus, Notifications & Queues

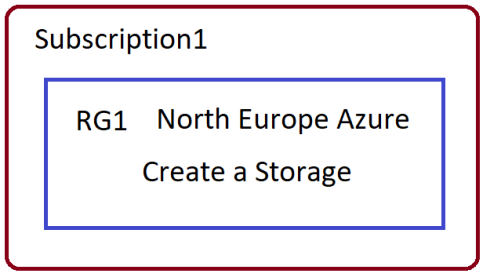
1. Routing Azure Event Grid Events to a Service Bus Queue – Made of Strings
2. Azure Event Grid documentation | Microsoft Learn
3. Compare Azure messaging services - Azure Event Grid | Microsoft Learn
4. What is Azure Notification Hubs? | Microsoft Learn
5. Compare Azure Storage queues and Service Bus queues - Azure Service Bus | Microsoft Learn
6. Azure Queue storage trigger for Azure Functions | Microsoft Learn
7. Quickstart - Use the Azure CLI to create a Service Bus queue - Azure Service Bus | Microsoft Learn
8. Compare Azure Storage queues and Service Bus queues - Azure Service Bus | Microsoft Learn

Automate, Workflow, Functions, Tasks

9. Automatically deploy code from Git-Hub to the newly created web app
10. ARM template & JSON
    - Variables in templates - Azure Resource Manager | Microsoft Learn
    - Tutorial - add variable to template - Azure Resource Manager | Microsoft Learn
    - 

    - Birds Eye ARM Templates

      Subscription named Subscription1
      Resource Group name RG1 in in North Europe Azure region.
      Create a Storage account in RG1 with same location as RG1

    -

11. Add regions, change failover priority, trigger failover for an Azure Cosmos DB account | Microsoft Learn
12. Azure Functions trigger and binding example | Microsoft Learn
    - In from the queue – out to the table
    - Azure Functions HTTP triggers and bindings | Microsoft Learn
    - Azure Functions trigger and binding example | Microsoft Learn
13. Logic Apps & Integration service environment (ISE)
    - Build approval-based automated workflows - Azure Logic Apps | Microsoft Learn
    - Logic Apps need access to secured resources, such as virtual machines (VMs) and other systems or services, that are inside or connected to an Azure virtual network you can create an integration service environment (ISE).
    - An ISE is an instance of the Logic Apps service that uses dedicated resources and runs separately from the "global" multi-tenant Logic Apps service.
14. Azure AD B2B collaboration overview - Microsoft Entra | Microsoft Learn
15. Overview - Access to Azure virtual networks - Azure Logic Apps | Microsoft Learn

Ess Q EL (SQL)

16. Create a API for NoSQL database and container for Azure Cosmos DB | Microsoft Learn
17. Cosmos DB and MongoDB
    - are NoSQL, or non-relational, databases.
    - Tutorial: Migrate MongoDB online to Azure Cosmos DB for MongoDB - Azure Database Migration Service | Microsoft Learn
18. query the records in the CosmosDB account using SQL queries,
19. Integration Service Environment
20. Announcing Azure Integration Service Environment for Logic Apps | Azure Blog | Microsoft Azure
    - Isolated and dedicated environment
    - Injected into your Azure virtual network, which allows you to deploy Logic Apps as a service on your VNET.
    - noisy neighbor issue
    - includes the free usage of 1 Standard Integration Account and 1 Enterprise connector.
21. space

Short definitions

1. A message is raw data (like orders in e commerce)
2. An event is a lightweight notification of a condition or a state change.
    a. **Event Grid** efficiently and reliably routes events from Azure and non-Azure resources. It distributes the events to registered subscriber endpoints.
    b. **Event Hub** is a service that provides a single solution that enables rapid data retrieval for real-time processing, and repeated replay of stored raw data. It can capture the streaming data into a file for processing and analysis.
    c. **Service Bus** is a brokered messaging system. It stores messages in a "broker" (for example, a queue) until the consuming party is ready to receive the messages.

Although they have some similarities each service is designed for particular scenarios.

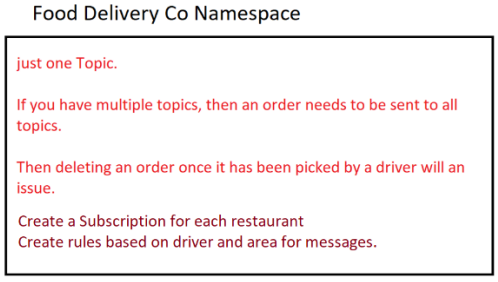
- An e-commerce site can use Service Bus to process the order
- Event Grid to respond to events like an item was shipped

For example, an e-commerce site can use Service Bus to process the order, Event Hubs to capture site telemetry, and Event Grid to respond to events like an item was shipped.

Longer discussions

Azure offers three services that assist with delivering events or messages throughout a solution.

- Azure Event Grid (app events)
  - Enables event-driven, reactive programming
  - Lowers costs by eliminating the need for constant polling.
  - Distributes the events to registered subscriber endpoints.
  - Has information to react to changes in services and applications.
  - Isn't a data pipeline, and doesn't deliver the actual object that was updated.
- Azure Event Hubs (data)
  - This service provides a single solution that enables
    - Rapid data retrieval for real-time processing
    - Repeated replay of stored raw data.
  - It can capture the streaming data into a file for processing and analysis.
- Azure Service Bus
  - Enterprise message broker with message queues and publish-subscribe topics.
  - The service is intended for enterprise applications that require transactions, ordering, duplicate detection, and instantaneous consistency.
  - Provide reliable state transition management for business processes.
  - When handling high-value messages that can't be lost or duplicated.



  - A driver selects the restaurants for which they will deliver orders.
    since the driver needs to choose the restaurant, that means the driver can be a subscriber.
  - Orders are sent to all available drivers in an area.
  - Only orders for the selected restaurants will appear for the driver.
  - The first driver to accept an order removes it from the list of available orders.
- Azure Notification Hubs (sending notifications)
  - Provide an easy-to-use and scaled-out push engine that enables you to send notifications to any platform (iOS, Android, Windows, etc.)
  - From any back-end (cloud or on-premises).
  - Platform Notification Systems (PNSes).

    - Offer basic push functionalities to deliver a message to a device with a provided handle, and have no common interface.
  - Telemetry is the automatic measurement and wireless transmission of data from remote sources.

implement push notifications

```csharp
static void ReceiveMessageAndSendNotification(string connectionString)
{
    // Initialize the Notification Hub
    string hubConnectionString = CloudConfigurationManager.GetSetting
            ("Microsoft.NotificationHub.ConnectionString");
    hub = NotificationHubClient.CreateClientFromConnectionString
            (hubConnectionString, "enterprisepushservicehub");

    SubscriptionClient Client =
        SubscriptionClient.CreateFromConnectionString
                (connectionString, sampleTopic, sampleSubscription);

    Client.Receive();

    // Continuously process messages received from the subscription
    while (true)
    {
        BrokeredMessage message = Client.Receive();
        var toastMessage = @"<toast><visual><binding template=""ToastText01""><text id=""1"">{messag

        if (message != null)
        {
            try
            {
                Console.WriteLine(message.MessageId);
                Console.WriteLine(message.SequenceNumber);
                string messageBody = message.GetBody<string>();
                Console.WriteLine("Body: " + messageBody + "\n");

                toastMessage = toastMessage.Replace("{messagepayload}", messageBody);
                SendNotificationAsync(toastMessage);
```

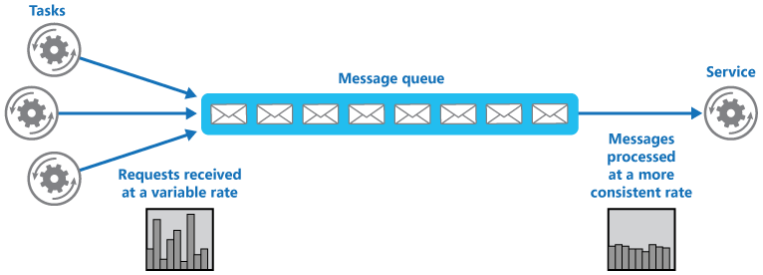
  -
  - Notification Hubs enterprise push architecture | Microsoft Learn

Things that might get monitored

  - Sensors at the source measure either electrical data, such as voltage and current, or physical data, such as temperature and pressure.
  - Create an Azure notification hub using the Azure portal | Microsoft Learn
  - **Send notifications to Universal Windows Platform apps using Azure Notification Hubs | Microsoft Learn

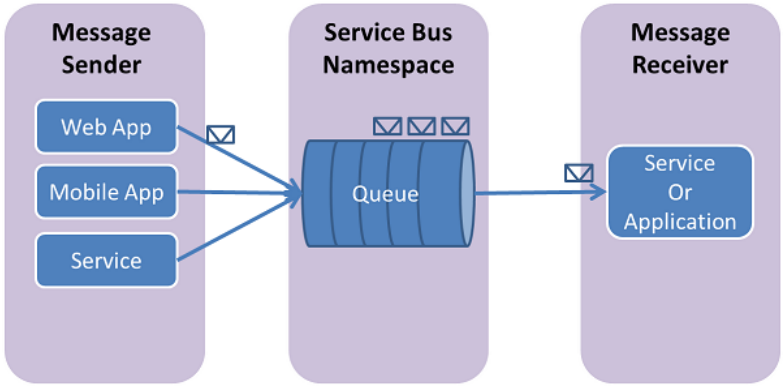
Queues

- Storage queues.
  - Allow you to store large numbers of messages.
  - Access messages from anywhere in the world via authenticated calls using HTTP or HTTPS.
  - Your application must store over 80 gigabytes of messages in a queue.
  - Your application wants to track progress for processing a message in the queue. It's useful if the worker processing a message crashes. Another worker can then use that information to continue from where the prior worker left off.
    - The queue storage trigger runs a function as messages are added to Azure Queue storage.
    - Use the queue trigger to start a function when a new item is received on a queue.
  - You require server-side logs of all of the transactions executed against your queues.
  - Standard queuing scenarios

- Decoupling application components to increase scalability and tolerance for failures
- Load leveling
  - Queue-Based Load Leveling pattern - Azure Architecture Center | Microsoft Learn
  - The queue decouples the tasks from the service, and the service can handle the messages at its own pace regardless of the volume of requests from concurrent tasks.
  - 

- Building process workflows.

- Azure Service Bus Topic example – who is the Topic & Subscriber?
  - a food delivery-based company (Topic)
    - – A driver (Subscriber) selects the restaurants for which they will deliver orders.
  - Creating a namespace for each restaurant would just be a maintenance overhead and difficult to keep track via a program.
  - Here since the driver needs to choose the restaurant, that means the driver can be a subscriber.
  - Here you should have just one Topic.
    - If you have multiple topics, then an order needs to be sent to all topics. Then deleting an order once it has been picked by a driver will an issue. So, Option A gets ruled out.
  - You can create subscriptions and create rules based on driver and area.
- Azure Service Bus queues
  - Queuing, publish/subscribe
  - Receive messages without having to poll the queue by using long-polling
  - Requires guaranteed first-in-first-out (FIFO) ordered delivery.
  - Need to support automatic duplicate detection.
  - Requires transactional behavior and atomicity when sending or receiving multiple messages
  - RBAC model to the queues with different rights/permissions for senders and receivers.
  - "At-Most-Once" and the "At-Least-Once" delivery
  - Use the Azure portal to create a Service Bus queue - Azure Service Bus | Microsoft Learn



What is the distinction between services that deliver an event and services that deliver a message?

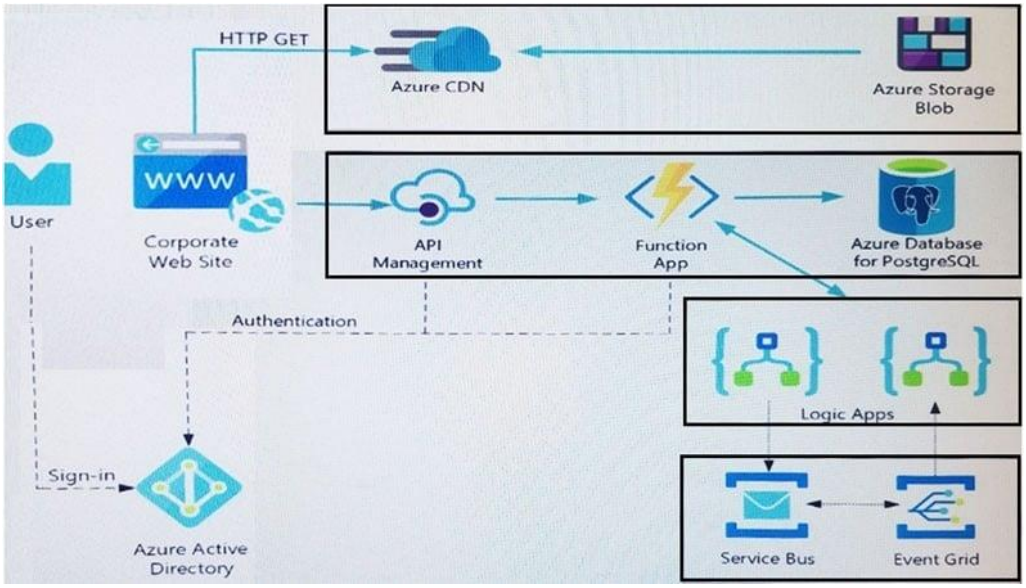
- Event:
  - An event is a notification of a condition or a state change.
  - The event data has information about what happened but doesn't have the data that triggered the event.
  - For example, an event notifies consumers that a file was created. It may have general information about the file, but it doesn't have the file itself.
  - Discrete events are ideal for serverless solutions that need to scale.
  - A series of events reports a condition and are analyzable.
  - The events are time-ordered and interrelated.
  - The consumer needs the sequenced series of events to analyze what happened.
- Message
  - A message is raw data produced by a service to be consumed or stored elsewhere.
  - The message contains the data that triggered the message pipeline.
  - The publisher of the message has an expectation about how the consumer handles the message.
  - A contract exists between the two sides.
  - For example, the publisher sends a message with the raw data, and expects the consumer to create a file from that data and send a response when the work is done.
- space

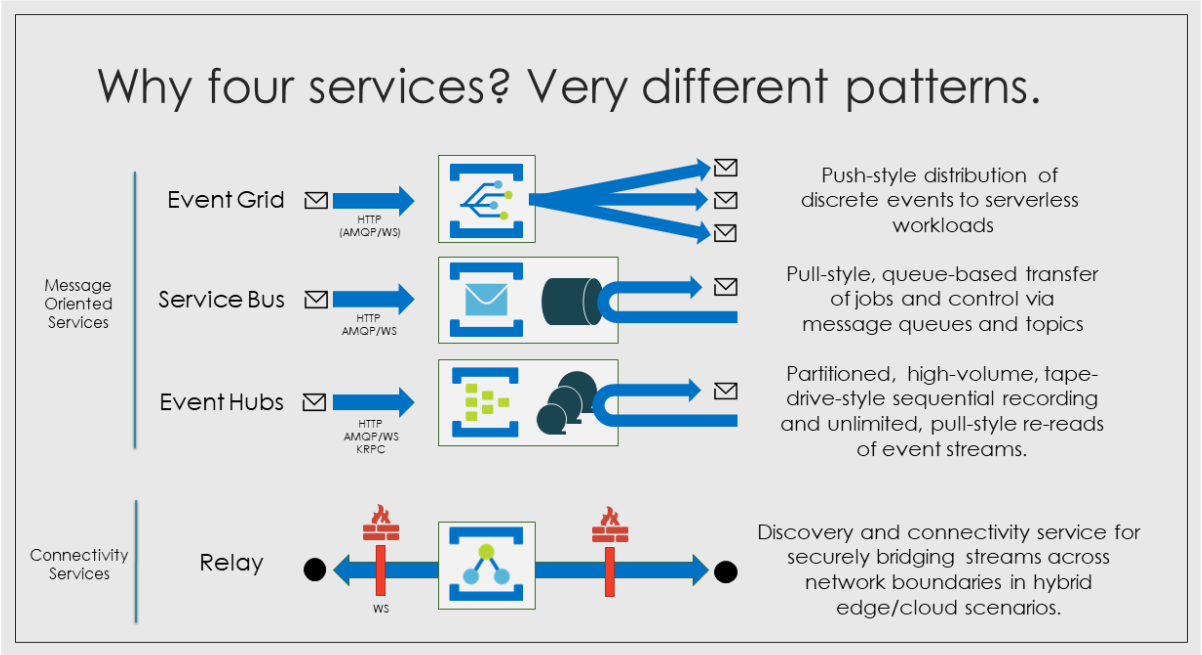
Messaging diffs

Event hub is a managed service that helps ingest, process, and monitor events think data pipelines for real-time analytics.

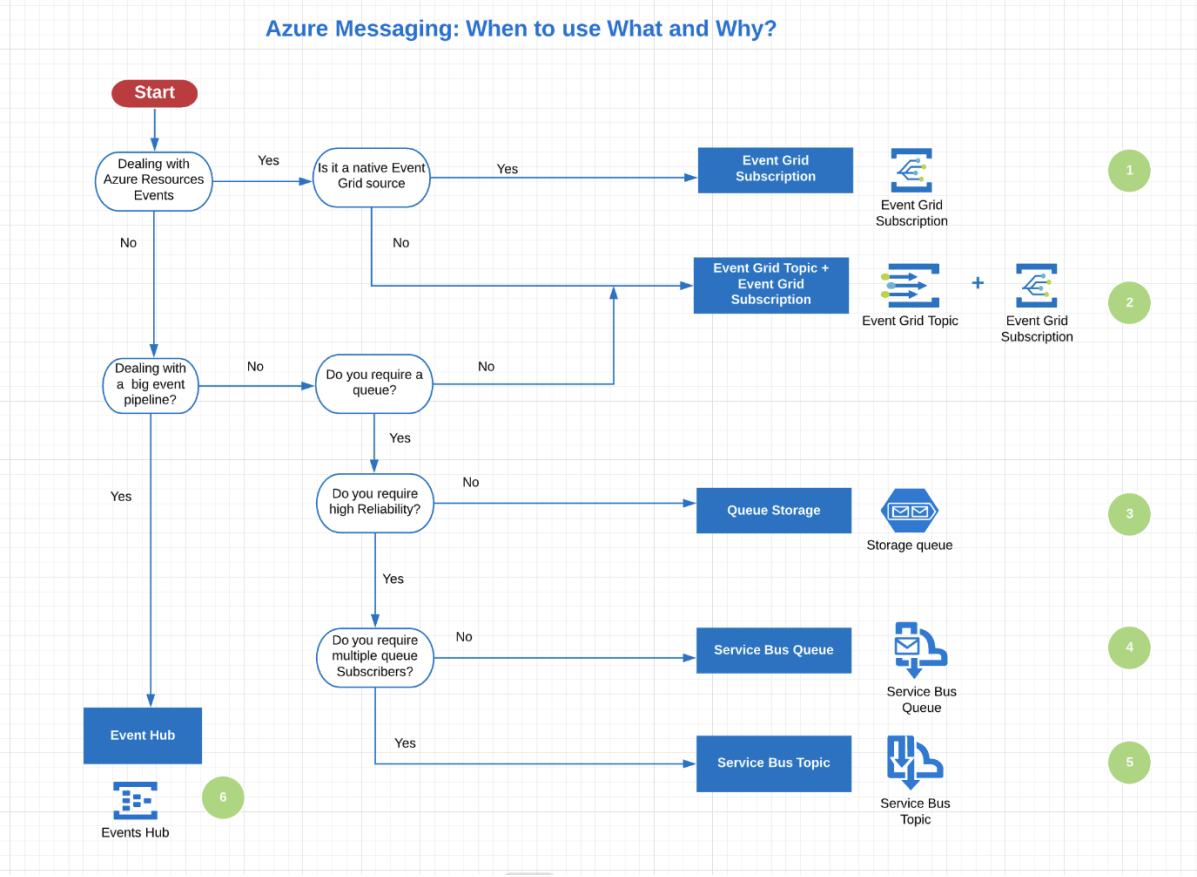
Event routing service that allows you to easily publish and subscribe to events

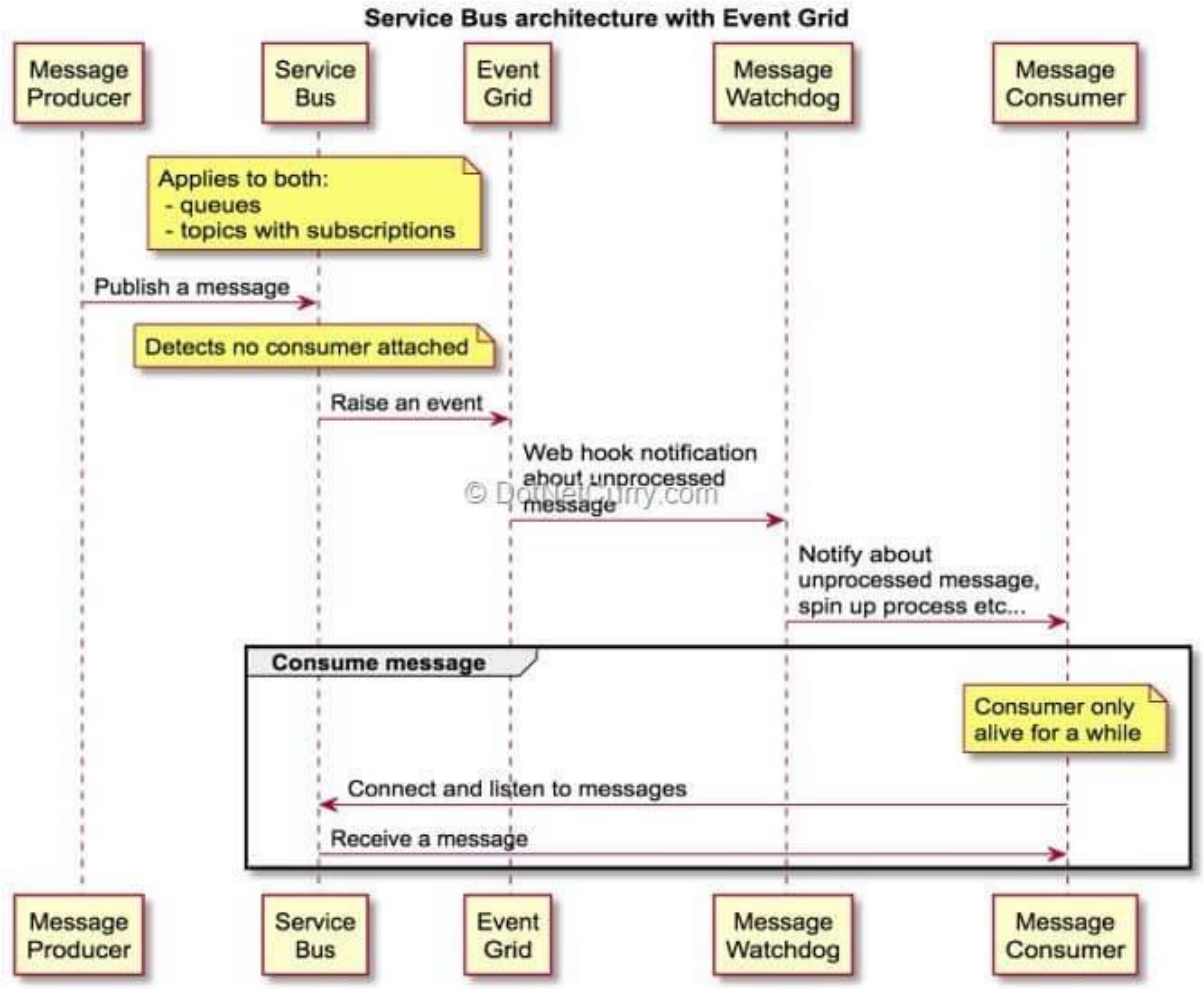
Both Service Bus and Event Grid support the Dead-letter queues. The only difference is that in Service Bus Dead-letter queue is available by default whereas in Event Grid it needs to be configured using a Storage account.


HTTP GET
Azure CDN
Azure Storage Blob
User
www
Corporate Web Site
API Management
Function App
Azure Database for PostgreSQL
Authentication
Logic Apps
Sign-in
Azure Active Directory
Service Bus
Event Grid


Why four services? Very different patterns.
Message Oriented Services
Event Grid
HTTP (AMQP/WS)
Push-style distribution of discrete events to serverless workloads
Service Bus
HTTP AMQP/WS
Pull-style, queue-based transfer of jobs and control via message queues and topics
Event Hubs
HTTP AMQP/WS KRPC
Partitioned, high-volume, tape-drive-style sequential recording and unlimited, pull-style re-reads of event streams.
Connectivity Services
Relay
WS
Discovery and connectivity service for securely bridging streams across network boundaries in hybrid edge/cloud scenarios.

Service	Type	Purpose	Examples
Azure Event Hubs	Event streaming (series)	Big data steaming pipeline	Telemetry and distributed data streaming; streaming log analytics
Azure Event Grid	Event distribution (discrete)	Reactive programming	Trigger Azure Functions when a file delivered; react to status changes
Azure Service Bus	Message	Enterprise application messaging	Order processing; financial transactions; FIFO message queues; AMQP





Azure Service Bus

Messaging

Transactions

Control

Pull

Azure Event Hubs

Series Events

Telemetry

Streaming

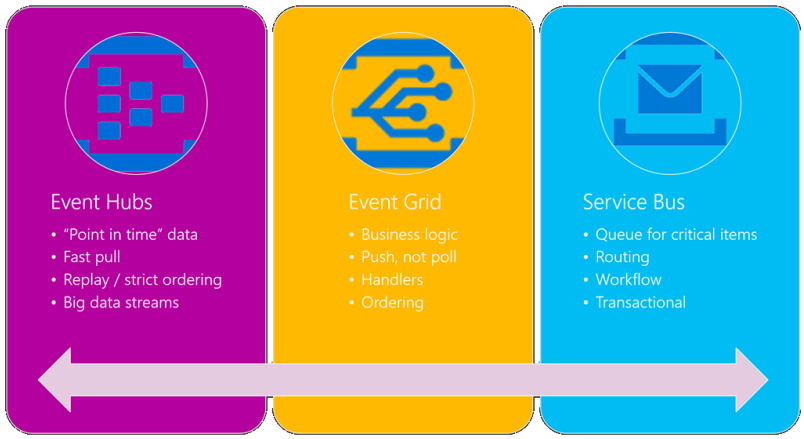
Azure Event Grid

Discrete Events

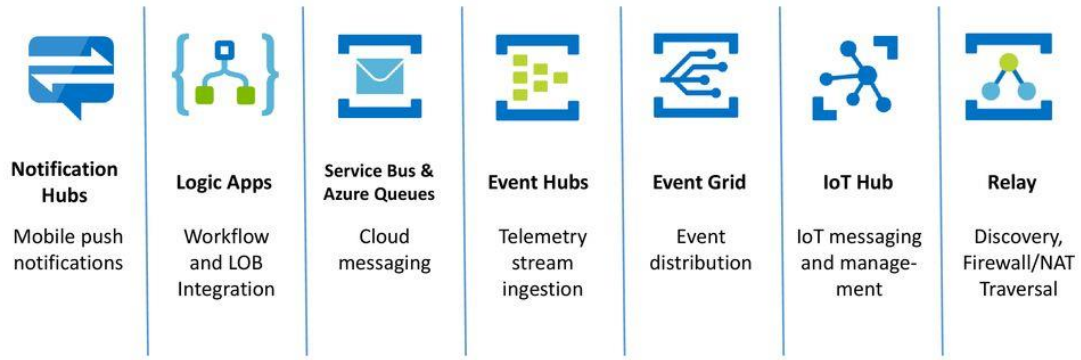
Event Handling

Reactive

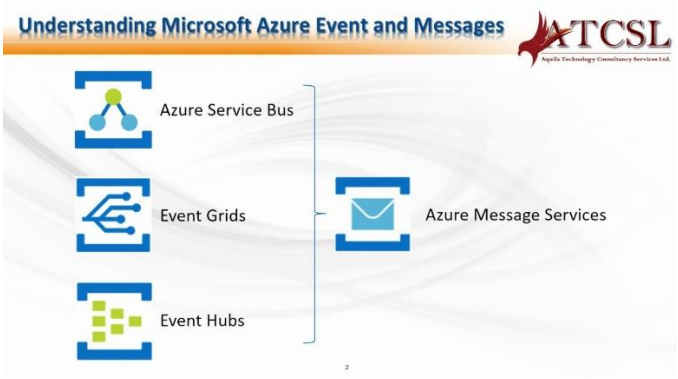
Push

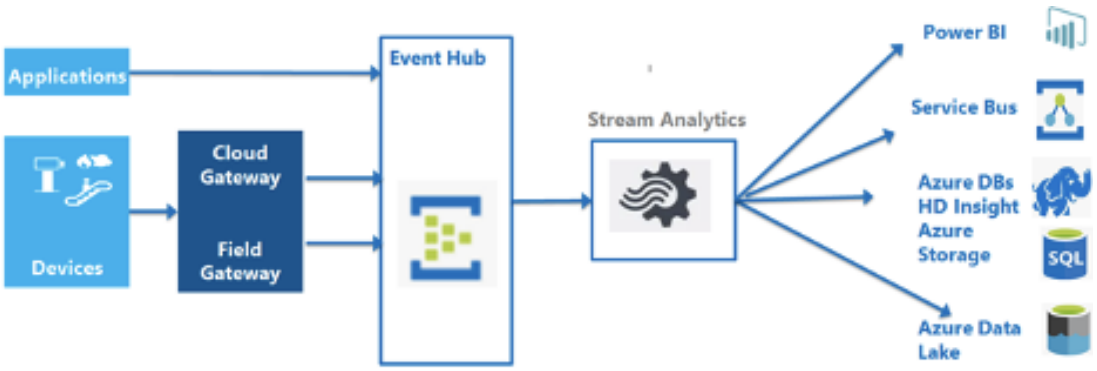


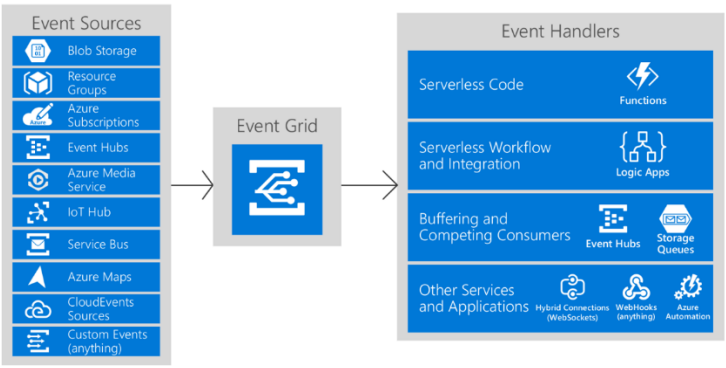
The Azure messaging landscape



codit



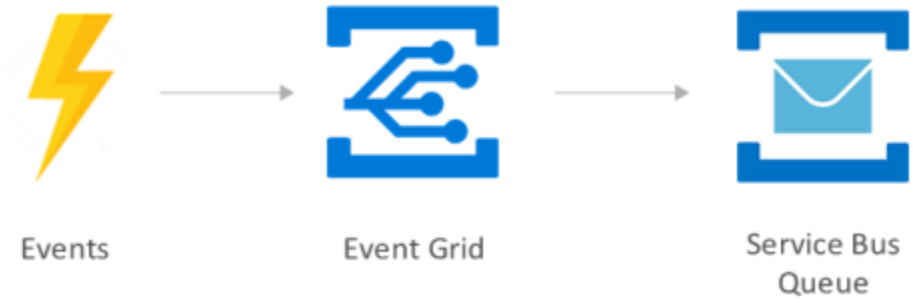




Event Publishers Event Handlers

1. Event Grid one of the coolest (and most innovative) services on Azure is it's unique integration between event sources and event handlers.
2. Event sources can emerge from a continually growing list of Azure services. A source can also be a custom event that originates from any line of business application or service, running anywhere.
3. Event handlers can be services on Azure, such as Azure Functions, Logic Apps, an automation run book and several more. In addition, webhooks (or callbacks) expand Event Grid's reach by supporting any reachable HTTPS endpoint as a handler.

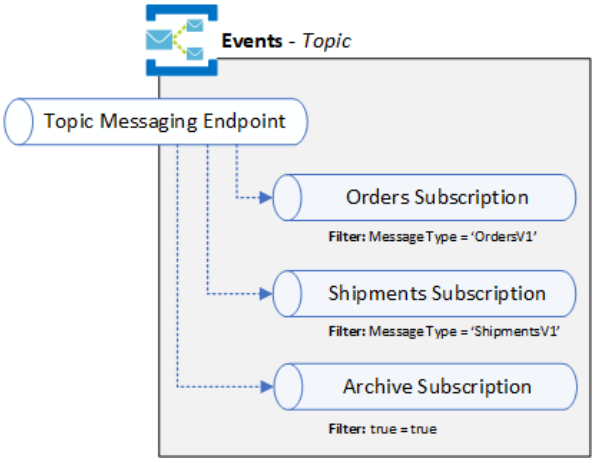
4. Delivery of events are supported by a configurable retry schedule, filters and dead-letter channel options. If you're interested in learning more about Event Grid delivery, please see this MSDN article: Managing Event Delivery with Azure Event Grid.
5. I really like this updated diagram of Event Grid because it relates handlers into categories (workflows, serverless, etc.) that are typically associated with other, commonly related requirements.

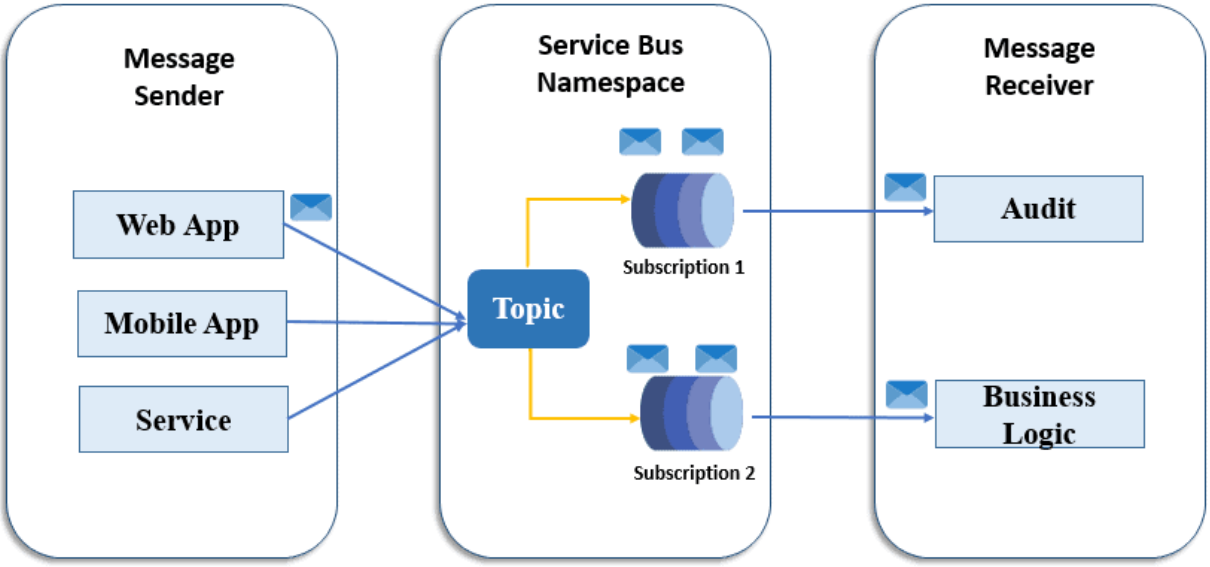


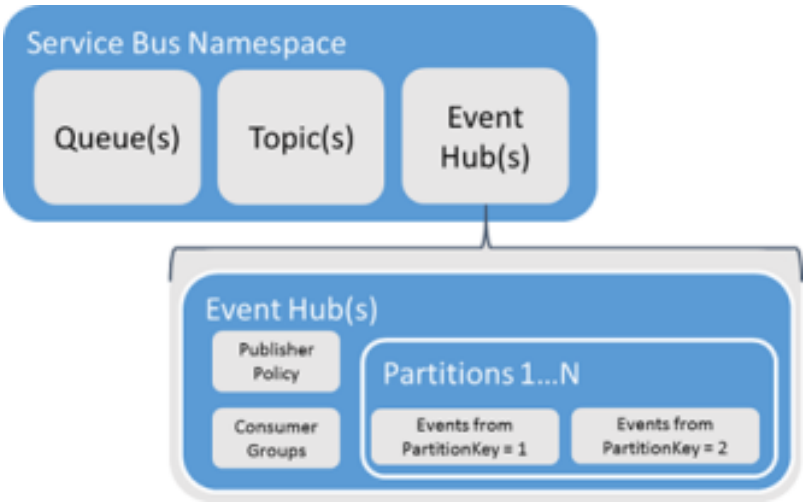
6.

Subscribing to events with a queue offers several benefits:

7. Queue-based load leveling. Leveraging a queue can not only decouple tasks but also offer a buffer for back-end services that otherwise might be overwhelmed with an influx of messages. With a queue in place, those services can now dequeue and process those messages at a responsible rate.
8. Long running tasks. Event Grid expects to receive an acknowledgment from it's handler fairly quickly (a 200 OK). If it does not receive a response in a timely manner, it will resend the event based on the configured retry schedule. In some cases, this could result in multiple events being sent to a handler, and sometimes even ending up in a dead-letter channel if it never completes on time. Employing a queue as a handler will allow another process to come along and take the time it needs to complete the next steps.
9. Wire tap. I sometimes use this pattern (with other pub/sub services such as Service Bus topics) to inspect messages when troubleshooting and monitoring a solution. It also comes in handy in testing scenarios when you simply want to review the events as they are coming in.
10. Other
11. Enterprise Integration Patterns - Wire Tap
12. Queue-Based Load Leveling pattern - Azure Architecture Center | Microsoft Learn







The token is a text string, included in the request header.

[Making Authenticated Requests - OAuth 2.0 Simplified]

The access token is not intended to be parsed or understood by your application. The only thing your application should do with it is use it to make API requests.

Some services will use structured tokens like JWTs as their access tokens, described in Self-Encoded Access Tokens but the client does not need to worry about decoding the token in this case.

[Self-Encoded Access Tokens - OAuth 2.0 Simplified]

POST /resource/1/update HTTP/1.1

Authorization: Bearer RsT5OjbzRn430zqMLgV3Ia"

Host: api.authorization-server.com

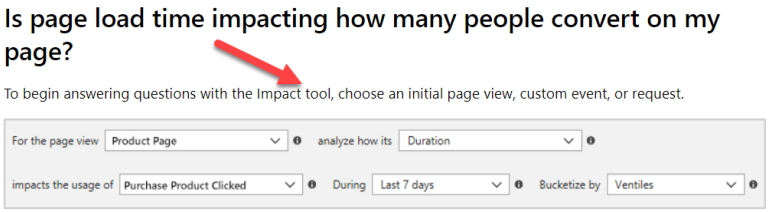
description=Hello+World

Insights

1. Application Insights Profiler
2. Application Insights
    a. Diagnose with Live Metrics - Application Insights - Azure Monitor | Microsoft Learn – 12/06/22
    b. Track custom operations with Application Insights .NET SDK - Azure Monitor | Microsoft Learn
    c. Telemetry.Id & Telemetry.Context.Operation.Id

    Typical things to look for:

    d. Funnel
        i. How they are using your app.
        ii. Application Insights funnels - Azure Monitor | Microsoft Learn
    e. Impact
        i. Is page load time impacting how many people convert on my page

        ii. 

        iii. Application Insights usage impact - Azure Monitor | Microsoft Learn
    f. Retention
        i. How many return
        ii. Analyze web app user retention with Application Insights - Azure Monitor | Microsoft Learn
    g. User Flows
        i. Application Insights User Flows analyzes navigation flows - Azure Monitor | Microsoft Learn

ii. User Flows tool visualizes how users move between the pages and features of your site.
- How do users move away from a page on your site?
- What do users select on a page on your site?
- Where are the places that users churn most from your site?
- Are there places where users repeat the same action over and over?

h. space

3. Aka QuickPulse
4. Control message routing – what is message payload? JSON?

Azure Kubernetes

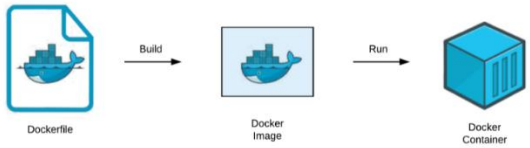
5. Sticky sessions or session affinity, is a feature that allows you to keep a session alive for a certain period of time.
6. In a Kubernetes cluster, all the traffic from a client to an application, even if you scale from 1 to 3 or more replicas, will be redirected to the same pod.
7. Annotations - NGINX Ingress Controller (kubernetes.github.io)
8. In some cases, you may want to "canary" a new set of changes by sending a small number of requests to a different service than the production service. The canary annotation enables the Ingress spec to act as an alternative service for requests to route to depending on the rules applied.
9. Why Consumption Azure Functions create an App Service Plan and Storage Account? - Stack Overflow
10. Azure Instance Metadata Service for Windows - Azure Virtual Machines | Microsoft Learn
11. Just-in-time virtual machine access in Microsoft Defender for Cloud | Microsoft Learn

Azure Containers

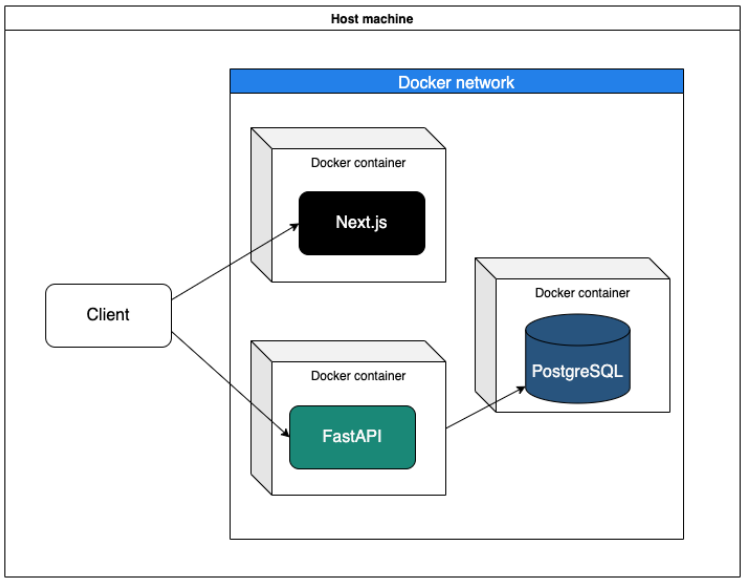
12. Windows container base images | Microsoft Learn
13. What is a container? | Microsoft Azure

Docker

14. Dockerfile & Linux file system
15. A Beginner's Guide to Building a Docker Image of Your Node.js Application - Maxim Orlov
16. Why do we put the application in /usr/src/app you're wondering? This is what Docker suggests and it follows the Linux Filesystem Hierarchy. It doesn't matter much in which directory you store your application code but it's better to stick to the recommendation for consistency and avoid overwriting files or directories used by the OS.
17. COPY package*.json ./ - Copies package.json (and package-lock.json if it exists) into the image
18. The COPY instruction does exactly what it says. It copies your application's package.json and package-lock.json files from the host file system to the present location (./) in your image. Which is in this case /usr/src/app as we've defined in the previous step. COPY takes two arguments: source and destination. Source is relative to the location of Dockerfile in your application. Destination is relative to WORKDIR.



19.



20.

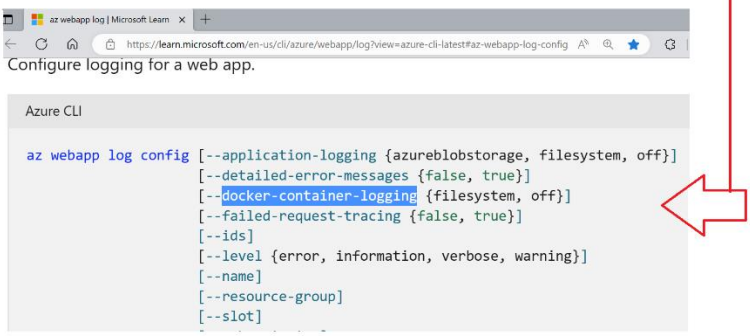
You need to complete the following Azure CLI script for this

az webapp log config *—name skillcertlabwebapp —resource-group skillcertlab-rg* Slot 2

filesystem

slot 2 could be from the list of items

az Slot 3 *log* Slot 4 *—name skillcertlabwebapp —resource-group skillcertlab-rg*

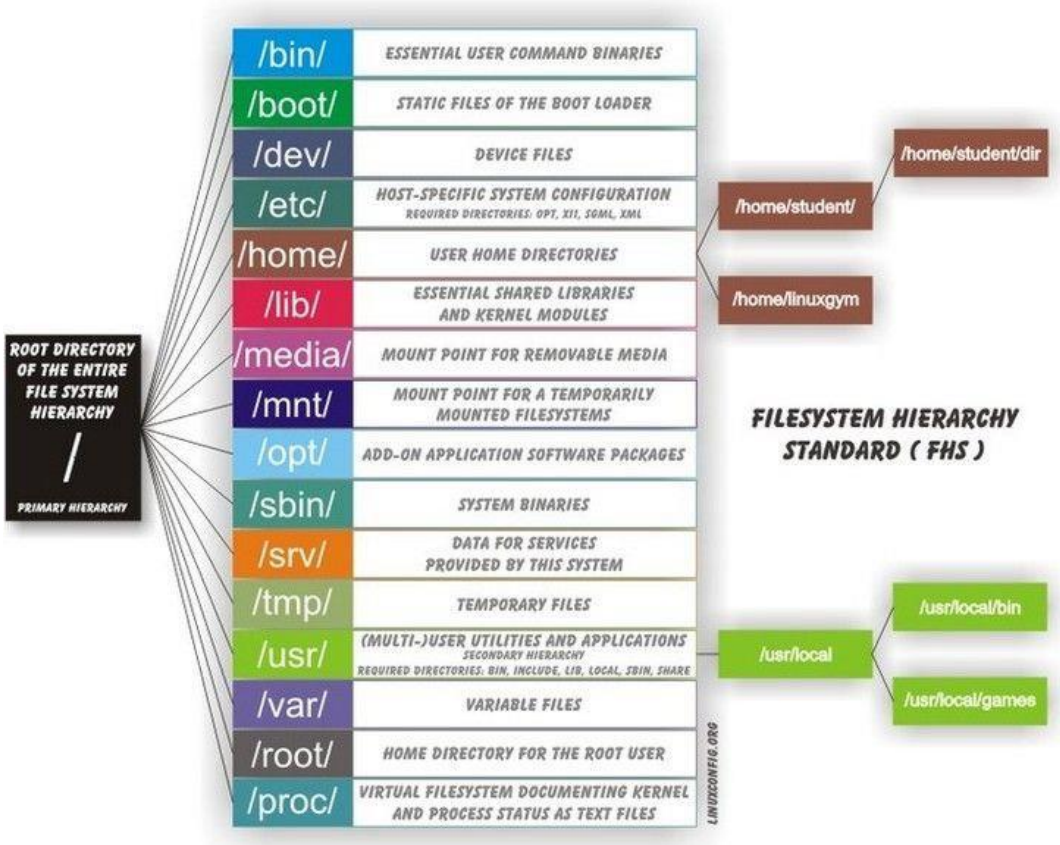


21.

22. Docker Container Logging
    a. az webapp log | Microsoft Learn –
    b. –docker-container-logging
23. Azure Container Registry Service

24. OpenAPI
25. groupMembershipClaims
    a. [OAuth 2.0 implicit grant flow - The Microsoft identity platform - Microsoft Entra | Microsoft Learn]
26. oauth2AllowImplicitFlow
    a. Specifies whether this web app can request OAuth2.0 implicit flow access tokens. The default is false.
27. headless authentication
    a. Headless Identity APIs help you separate back-end authentication processes from front-end identity experiences.
    b. [asp.net - Headless authentication with Azure AD (user/pass combination) - Stack Overflow]
    c. [Registry authentication options - Azure Container Registry | Microsoft Learn]
    d. [asp.net - Headless authentication with Azure AD (user/pass combination) - Stack Overflow]
    e. Authenticate to a registry directly via individual login
    f. Applications and container orchestrators can perform unattended, or "headless," authentication by using an Azure Active Directory (Azure AD) service principals.
    g. [Authenticate with Azure AD for access - Azure Data Explorer | Microsoft Learn]
    h. main authenticating scenarios are:
        i. A client application authenticating a signed-in user. In this scenario, an interactive (client) application triggers an Azure AD prompt to the user for credentials (such as username and password). See user authentication,
        ii. A "headless" application. In this scenario, an application is running with no user present to provide credentials. Instead the application authenticates as "itself" to Azure AD using some credentials it has been configured with. See application authentication.
        iii. On-behalf-of authentication. In this scenario, sometimes called the "web service" or "web app" scenario, the application gets an Azure AD access token from another application, and then "converts" it to another Azure AD access token that can be used with Azure Data Explorer. In other words, the application acts as a mediator between the user or application that provided credentials and the Azure Data Explorer service. See on-behalf-of authentication.
28. Manifest file
    a.
29. .deployment file
    a. Allow you to specify a project or folder to be deployed. It has to be at the root of the repository and it's in .ini format.
    b.

```
[config]
command = deploy.cmd
```



30.
31. [Including a space in a path name in a Dockerfile | ASP.NET Monsters (aspnetmonsters.com)]

Staging Environments

1. [Set up staging environments - Azure App Service | Microsoft Learn]
2. Auto Swap
3. Auto swap streamlines Azure DevOps scenarios where you want to deploy your app continuously with zero cold starts and zero downtime for customers of the app. When auto swap is enabled from a slot into production, every time you push your code changes to that slot, App Service automatically swaps the app into production after it's warmed up in the source slot.
4. [Scale Azure Cosmos DB on a schedule by using Azure Functions timer | Microsoft Learn]
5. [Application Initialization <applicationInitialization> | Microsoft Learn]
6. [Azure built-in roles - Azure RBAC | Microsoft Learn]
7. 34. Question

   You need to create an Azure key vault. The solution must ensure that any object deleted from the key vault be retained for 90 days.

   Which two parameters should you add to complete below command?

   New-AzKeyVault -Name MyKeyVault -ResourceGroupName RG1 -Location westus

   - [ ] #NAME?
   - [x] EnablePurgeProtection
   - [ ] #NAME?
   - [ ] #NAME?
   - [ ] #NAME?

Microsoft Graph API

1. Register your app with the Azure AD v2.0 endpoint - Microsoft Graph | Microsoft Learn
    a. This step creates automatically generated Application (client) ID and must be done first – then you can create or add an app using the ID.
2. Microsoft Authentication Library (MSAL) enables developers to acquire tokens from the Microsoft identity platform in order to authenticate users and access secured web APIs.
    a. You can update user profile information using Graph API.
3.

Azure AD: Change Management Simplified - Microsoft Community Hub – 03/2022

Code Talk

RHH (Rabbit Hole Handling)

Everything is a container in RAM memory – create the container and fill it with stuff – Do C.R.U.D. with Text!

Relay races are common in running, orienteering, swimming, cross-country skiing, biathlon, or ice skating (usually with a baton carried in the hand).

In OOP - Relational Algebra

- The baton is a variable
- The handoff or "glue" that passes values around are things like:
    - Connection strings
    - Parameters inside functions(p1, p2),
    - DOM HTML tags, classes, id's

Tools for execution of commands through a terminal using interactive command-line prompts or a script vs. writing an app in C# or JavaScript.

Code, Coding, Scripting – what gets used where?

Scripting languages for Workflow, Tasks
vs. Object Oriented Programing an app...

Command	Azure CLI	Azure PowerShell
Create Resource Group	az group create --name <ResourceGroupName> --location eastus	New-AzResourceGroup -Name <ResourceGroupName> -Location eastus

-Use PowerShell to install Azure CLI to make the AZ (Bash) commands run

Azure PowerShell

```
Get-Command -Module Az.*
```

- cmd.exe on Windows
- Bash on Linux or macOS

-CLI commands start with az

Azure CLI

```
az webapp --help
```

-dotnet new

.NET CLI

```
dotnet build-server shutdown -h|--help
dotnet build-server -h|--help
```

1. -Navigate folders with cd, dir, md, ..

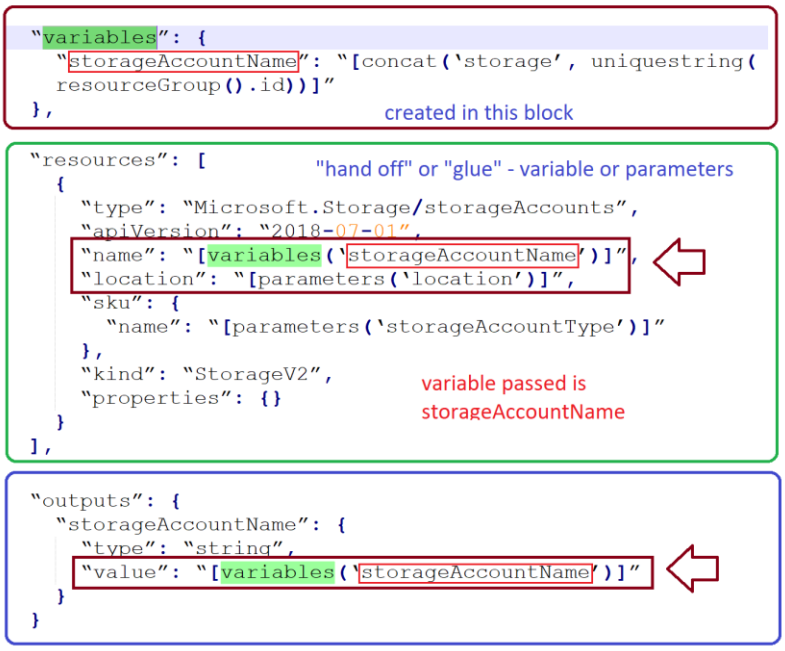
2. JavaScript — Fill in the blank Q - AZ-204 — Copy

```
function validateToDoItemTimestamp() {
    var context = getContext();
    var request = context.getRequest();

    // item to be created in the current operation
    var itemToCreate = request.getBody();

    // validate properties
    if (!("timestamp" in itemToCreate)) {
        var ts = new Date();
        itemToCreate["timestamp"] = ts.getTime();
    }

    // update the item that will be created
    request.setBody(itemToCreate);
}
```

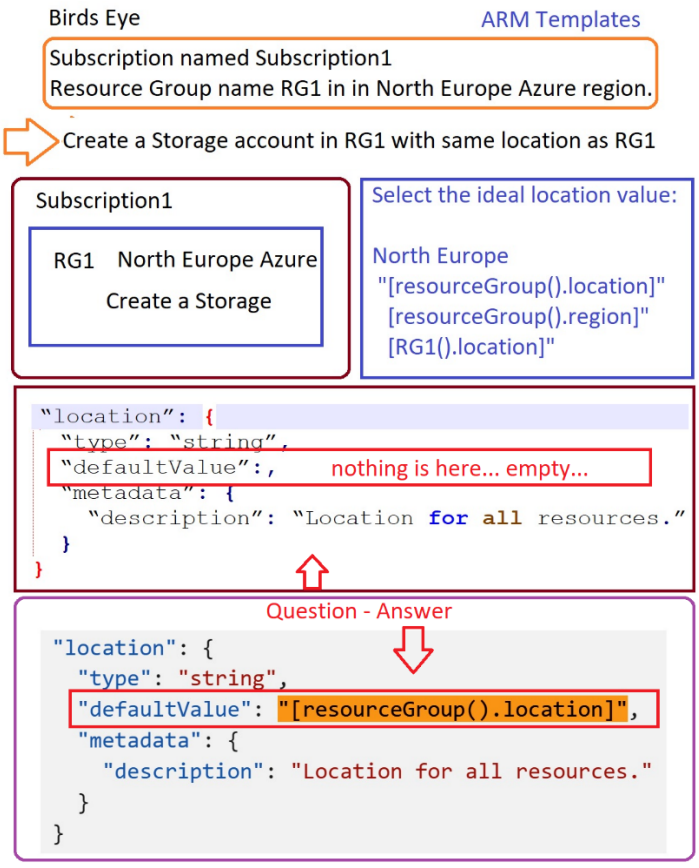
```
function OrderTip() {
var r= getContext().getRequest();

var is=r.getBody();

If(!("Ordertip" in i)) {
i[tip]=0;
}

r.setBody(i);

}
```

Object Oriented Programming in JSON - Passing Values w/variables & parameters

3.



4.



Azure CLI

5.
6. [CLI: Deploy an app from GitHub - Azure App Service | Microsoft Learn]

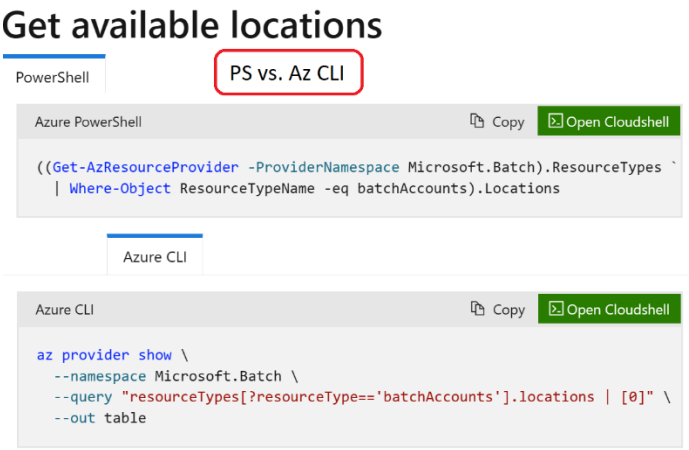
cmd.exe or PowerShell on Windows - Bash on Linux or macOS

- Azure Cloud Shell (runs AZ CLI & PowerShell)
- Terminal (runs Command Prompt, AZ CLI & PowerShell in separate tabs)
- Visual Studio Code (has built in Terminal)
- Command Prompt
- PowerShell
- WSL
    - Windows Subsystem for Linux (WSL) is a feature of Windows that allows developers to run a Linux environment
- [Windows Terminal SSH | Microsoft Learn]
    - a connectivity tool for remote sign-in (see also RDP for Windows)
- Azure Cloud Shell Connector

Understanding: Azure CLI, dotnet, C#, JSON, SQL, PowerShell, Bash

- Azure Command-Line Interface (CLI) is a cross-platform command-line tool to connect to Azure and execute administrative commands on Azure resources.
- Azure CLI has an installer that makes its commands executable in all four shell environments. Azure PowerShell is set of cmdlets packaged as a PowerShell module named Az ; not an executable.
- Windows PowerShell or PowerShell must be used to install the Az module.
- Azure Command-Line Interface (CLI) - Overview | Microsoft Learn
- .NET CLI | Microsoft Learn
- .NET command-line interface (CLI) is a cross-platform toolchain for developing, building, running, and publishing .NET applications. - dotnet new
- Create a project template for dotnet new - .NET | Microsoft Learn

Difference PowerShell vs. Azure CLI



- Quickstart - Set & retrieve a secret from Key Vault using PowerShell | Microsoft Learn
- Get-AzContext (Az.Accounts) | Microsoft Learn

store and retrieve a storage account key secret from Azure Key Vault.

which order

3 1.Get-AzStorageAccountKey -ResourceGroupName $resGroup -Name $storAcct
5 2.Get-AzKeyVaultSecret -VaultName $vaultName
1 3.Get-AzSubscription
4 4.$secretvalue = ConvertTo-SecureString $storAcctKey -AsPlainText -Force
Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue
2 5.Set-AzContext -SubscriptionId $subscriptionID

Commands fall into a logical order based on Azure parent/child - subscription contains resource group

-Cart and Horse -
-What must exist first?

3,5,1,4,2

The correct script is

PowerShell verb-noun Get-AzContext

1.Get-AzSubscription
2.Set-AzContext -SubscriptionId $subscriptionID
3.Get-AzStorageAccountKey -ResourceGroupName $resGroup -Name $storAcct
4.$secretvalue = ConvertTo-SecureString $storAcctKey -AsPlainText -Force
Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue
5.Get-AzKeyVaultSecret -VaultName $vaultName

AzContext: gets the Active Directory account, Active Directory tenant, Azure subscription, and the targeted Azure environment. (more "glue" to put parts together)

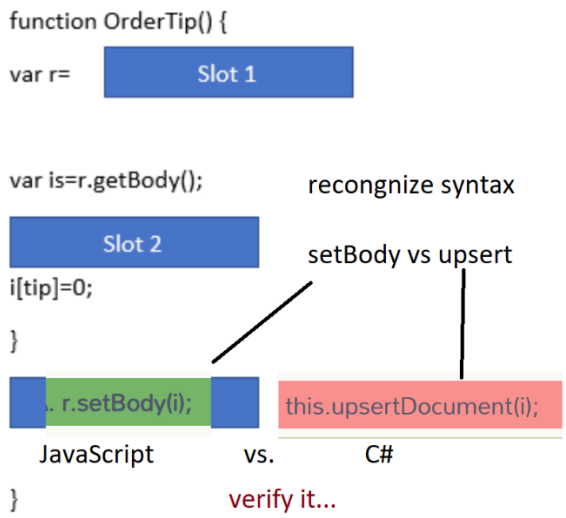
In the Context of - what (a person in a data model, an Azure Account here)

Base code exercises for GitHub 204 project

- How To Create A List In C#? (c-sharpcorner.com)
- C# List Tutorial - Everything You Need To Learn About List In C# (c-sharpcorner.com)

```
PS C:\GoHere\T_Labs\Projects\Azure_Tests\az204svcbus> dotnet new console
The command could not be loaded, possibly because:
  * You intended to execute a .NET application:
      The application 'new' does not exist.
  * You intended to execute a .NET SDK command:
      No .NET SDKs were found.

Download a .NET SDK:
https://aka.ms/dotnet-download

Learn about SDK resolution:
https://aka.ms/dotnet/sdk-not-found
PS C:\GoHere\T_Labs\Projects\Azure_Tests\az204svcbus>
```

dotnet library missing



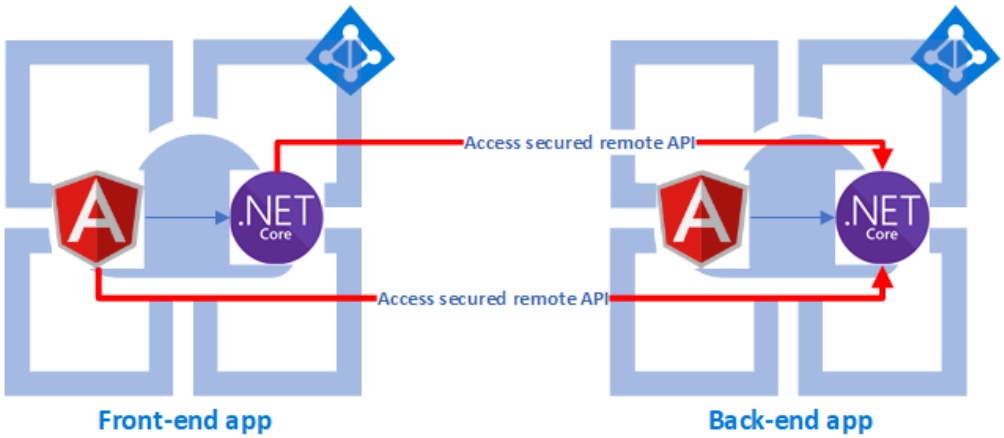
What does ModelBuilder entity do?

Returns an object that can be used to configure a given entity type in the model. If the entity type is not already part of the model, it will be added to the model.

ModelBuilder.Model Property (Microsoft.EntityFrameworkCore) | Microsoft Learn

Reference

3. Configure daemon apps that call web APIs - Microsoft Entra | Microsoft Learn
4. Get access without a user - Microsoft Graph | Microsoft Learn
5. Create a Microsoft Graph client - Microsoft Graph | Microsoft Learn
6. Migrate from Azure AD PowerShell to Microsoft Graph PowerShell. | Microsoft Learn
7. Rotation tutorial for resources with two sets of credentials | Microsoft Learn
8. Feature-based comparison of the Azure API Management tiers | Microsoft Learn
9. Compare Azure messaging services - Azure Event Grid | Microsoft Learn
10. Tutorial: Authenticate users E2E - Azure App Service | Microsoft Learn



11.
12. Tutorial: send email with Logic Apps - Azure App Service | Microsoft Learn – tweets
13. Enable diagnostics logging - Azure App Service | Microsoft Learn

Azure API Management Policy

14. Policies in Azure API Management | Microsoft Learn
15. Azure API Management policy samples | Microsoft Learn
16. Add caching to improve performance in Azure API Management | Microsoft Learn
17. Azure API Management advanced policies | Microsoft Learn
    a. Where in the code block do you type the instructions
    b. Azure API Management policy samples | Microsoft Learn

XML `inbound`, `backend`, `outbound`, and `on-error` Copy

```
<policies>
  <inbound>
    <!-- statements to be applied to the request go here -->
  </inbound>
  <backend>
    <!-- statements to be applied before the request is forwarded to
         the backend service go here -->
  </backend>
  <outbound>
    <!-- statements to be applied to the response go here -->
  </outbound>
  <on-error>
    <!-- statements to be applied if there is an error condition go here -->
  </on-error>
</policies>
```

    c.

Outbound policies	Description
Filter response content	Demonstrates how to filter data elements from the response payload based on the product associated with the request.
Set response cache duration	Demonstrates how to set response cache duration using maxAge value in Cache-Control header sent by the backend.
On-error policies	**Description**
Log errors to Stackify	Shows how to add an error logging policy to send errors to Stackify for logging.

All else are inbound - 2023

    d.
18. Policy: How to cache?
    a. <inbound>
        i. - rewrite-uri policy
        ii. cache-lookup
    b. <backend>
        i.
    c. <outbound>
        i.

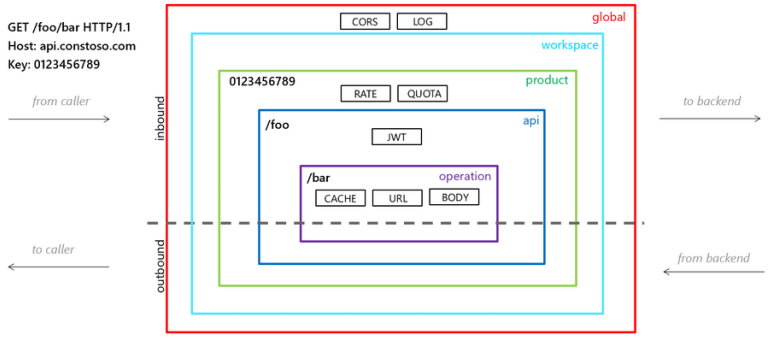
```
    <outbound>
        <!-- Update response body with user profile-->
        <find-and-replace
            from='"$userprofile$"'
            to="@((string)context.Variables["userprofile"])" />
        <base />
    </outbound>
</policies>
```

Azure API Management Policy

ii. 

d. <on-error>

e. **Policy scopes**



Event Hubs Capture

19. Capture streaming events - Azure Event Hubs - Azure Event Hubs | Microsoft Learn
    a. Azure Event Hubs enables you to automatically capture the streaming data in Event Hubs in an Azure Blob storage or Azure Data Lake Storage Gen 1 or Gen 2 account of your choice, with the added flexibility of specifying a time or size interval.
    b. Setting up Capture is fast, there are no administrative costs to run it, and it scales automatically with Event Hubs throughput units in the standard tier or processing units in the premium tier. Event Hubs Capture is the easiest way to load streaming data into Azure, and enables you to focus on data processing rather than on data capture.
    c. Apache Avro
    d. Capture Event Hubs data to ADLSG2 in parquet format | Microsoft Learn
    e. Stream Analytics no code editor
    f. Query captured data in Parquet format with Azure Synapse Analytics.
    g. Query using Azure Synapse Spark – Python
20. Key Vault
    a. Manage Azure Key Vault using CLI - Azure Key Vault | Microsoft Learn
    b. `az keyvault key create`
    c. `az keyvault key import`
    d. `az keyvault key list --vault-name "ContosoKeyVault"`
    e. `az keyvault secret list --vault-name "ContosoKeyVault"`
    f. `az keyvault certificate list --vault-name "ContosoKeyVault"`
    g. az keyvault | Microsoft Learn
21. Azure Data Factory
22. Hadoop

Azure Redis Cache (Redis is 3rd party)

23. Azure Cache for Redis Documentation | Microsoft Learn
    a. Redis improves the performance and scalability of an application that uses backend data stores heavily.
    b. It's able to process large volumes of application requests by keeping frequently accessed data in the server memory, which can be written to and read from quickly.

User session caching	This pattern is commonly used with shopping carts and other user history type information that a web application may want to associate with user cookies. Storing too much in a cookie can have a negative impact on performance as the cookie size grows and is passed and validated with every request. A typical solution is to use the cookie as a key to query the data in a backend database. Using an in-memory cache, like Azure Cache for Redis, to associate information with a user is much faster than interacting with a full relational database.

24. Ideally store the session state in Azure Redis.
25. Ensure that stale keys are removed from the cache.
    a. Choose an eviction policy
    b. Set a key expiration value.

> ## Memory management
>
> There are several things related to memory usage within your Redis server instance that you may want to consider. Here are a few:
>
> - **Choose an eviction policy that works for your application.** The default policy for Azure Redis is *volatile-lru*, which means that only keys that have a TTL value set will be eligible for eviction. If no keys have a TTL value, then the system won't evict any keys. If you want the system to allow any key to be evicted if under memory pressure, then you may want to consider the *allkeys-lru* policy.
>
> - **Set an expiration value on your keys.** An expiration will remove keys proactively instead of waiting until there's memory pressure. When eviction does kick in because of memory pressure, it can cause additional load on your server. For more information, see the documentation for the EXPIRE and EXPIREAT commands.

    c.
    d. TTL: Time to live
    e. Key eviction | Redis
    f. Policies : (recently & frequently)
        i. noeviction: New values aren't saved when memory limit is reached. When a database uses replication, this applies to the primary database
        ii. allkeys-lru: Keeps most recently used keys; removes least recently used (LRU) keys
        iii. allkeys-lfu: Keeps frequently used keys; removes least frequently used (LFU) keys
        iv. volatile-lru: Removes least recently used keys with the expire field set to true.
        v. volatile-lfu: Removes least frequently used keys with the expire field set to true.
        vi. allkeys-random: Randomly removes keys to make space for the new data added.
        vii. volatile-random: Randomly removes keys with expire field set to true.
        viii. volatile-ttl: Removes keys with expire field set to true and the shortest remaining time-to-live (TTL) value.
26. space

Automatic deployment of code from GitHub

1. Azure CLI | Copy | Try It

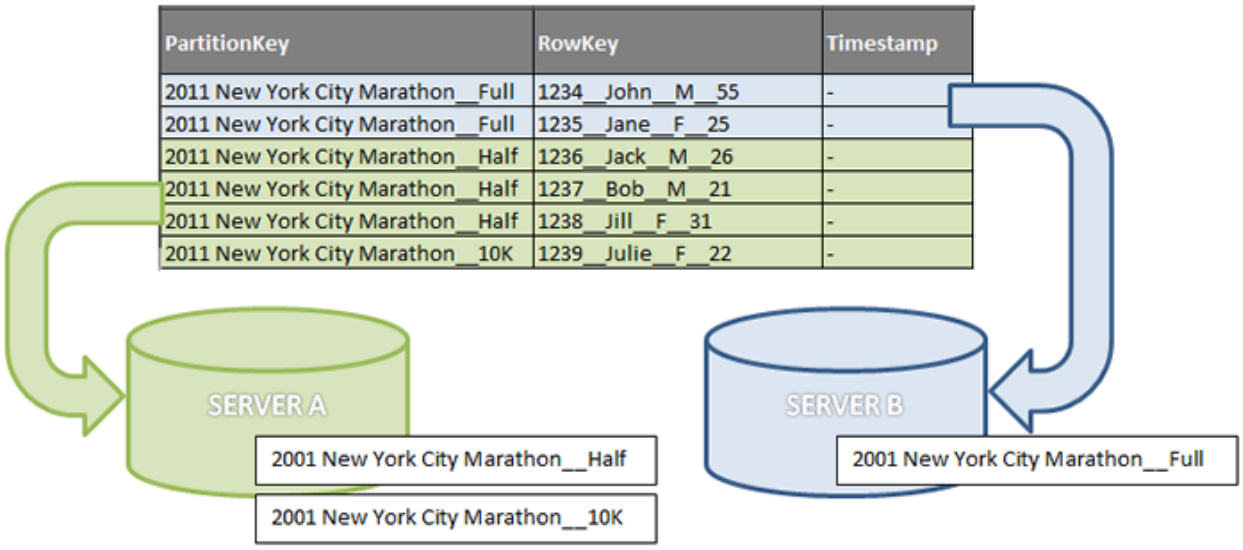
```
#!/bin/bash

# Replace the following URL with a public GitHub repo URL
gitrepo=https://github.com/Azure-Samples/php-docs-hello-world
webappname=mywebapp$RANDOM

# Create a resource group.
az group create --location westeurope --name myResourceGroup

# Create an App Service plan in `FREE` tier.
az appservice plan create --name $webappname --resource-group myResourceGroup --sku FREE

# Create a web app.
az webapp create --name $webappname --resource-group myResourceGroup --plan $webappname

# Deploy code from a public GitHub repository.
az webapp deployment source config --name $webappname --resource-group myResourceGroup \
--repo-url $gitrepo --branch master --manual-integration

# Copy the result of the following command into a browser to see the web app.
echo http://$webappname.azurewebsites.net
```

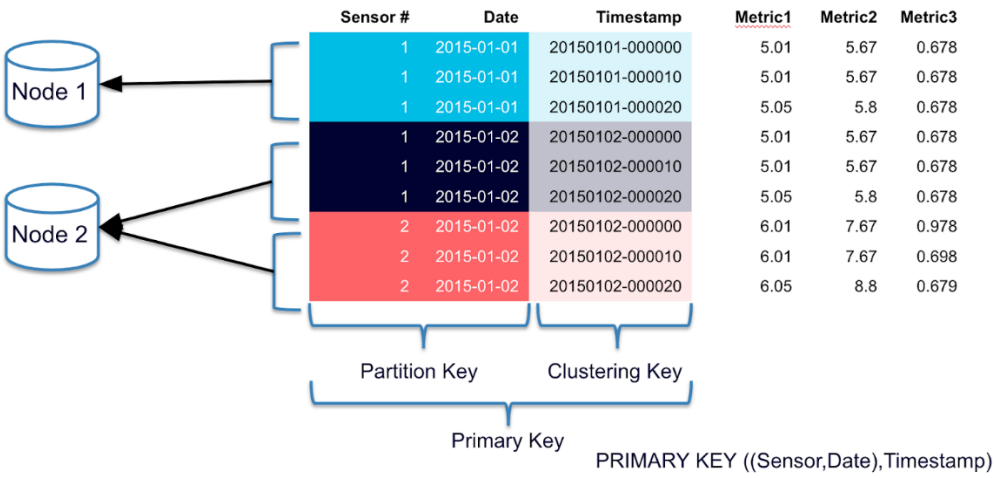
space

Table Talk

Table Storage: Partition and Row keys – Dividing or Grouping "Entity and Property"

[Azure Table Storage Design Manage And Scale Table Partitions (c-sharpcorner.com)]





Sensor #	Date	Timestamp	Metric1	Metric2	Metric3
1	2015-01-01	20150101-000000	5.01	5.67	0.678
1	2015-01-01	20150101-000010	5.01	5.67	0.678
1	2015-01-01	20150101-000020	5.05	5.8	0.678
1	2015-01-02	20150102-000000	5.01	5.67	0.678
1	2015-01-02	20150102-000010	5.01	5.67	0.678
1	2015-01-02	20150102-000020	5.05	5.8	0.678
2	2015-01-02	20150102-000000	6.01	7.67	0.978
2	2015-01-02	20150102-000010	6.01	7.67	0.698
2	2015-01-02	20150102-000020	6.05	8.8	0.679

Example Case Study

Current environment –

1. Windows Server 2016 virtual machine (VM) runs BizTalk Server 2016. The VM runs the following workflows:

2. 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:

3. Container API – This API provides container information including weight, contents, and other attributes.

4. Location API – This API provides location information regarding shipping ports of call and tracking stops.

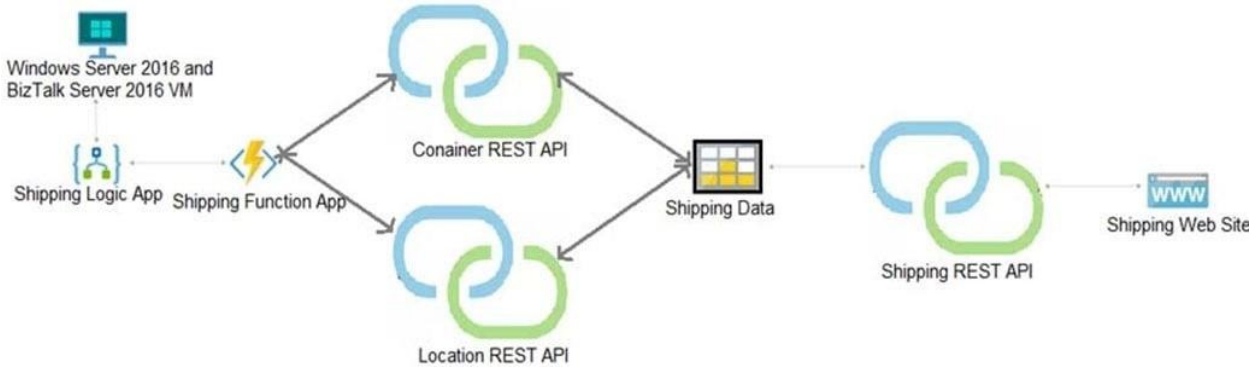
5. Shipping REST API – This API provides shipping information for use and display on the shipping website.

6. Shipping Data –
   The application uses MongoDB JSON document storage database for all container and transport information.

7. Shipping Web Site –
   The site displays shipping container tracking information and container contents. The site is located at http://shipping.wideworldimporters.com/

8. Proposed solution –
   The on-premises shipping application must be moved to Azure.
   a. The VM has been migrated to a new Standard_D16s_v3 Azure VM by using Azure Site Recovery to host BizTalk Server.
   b. Must remain running in Azure to complete the BizTalk component migrations.

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



10. Requirements –

11. Shipping Logic app –
    The Shipping Logic app must meet the following requirements:

    a. Support the ocean transport and inland transport workflows by using a Logic App.

    b. Support industry-standard protocol X12 message format for various messages including vessel content details and arrival notices.

    c. Secure resources to the corporate VNet and use dedicated storage resources with a fixed costing model.

    d. Maintain on-premises connectivity to support legacy applications and final BizTalk migrations.

12. Shipping Function app –

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

REST APIs –

13. 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.

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

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

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

Issues –

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

18. The following error message displays while you are testing the website:

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

Another section

I May Need To:

1. correct process of how the CDN and the Point of Presence (POP) server will distribute the image and list the items in the correct order.
2. e-commerce application. The application generates millions of orders and captures user behavior. Users must be informed as and when there is progress for their order. The application must support sales and marketing team to analyze live user behavior.
3. a hazard notification system that has a single signaling server which triggers audio and visual alarms to start and stop.
4. You implement Azure Service Bus to publish alarms. (partitions on device)(what ids get recorded)
5. Each alarm controller uses Azure Service Bus to receive alarm signals as part of a transaction. Alarm events must be recorded for audit purposes. Each transaction record must include information about the alarm type that was activated. You need to implement a reply trail auditing solution.
6. ensure that authentication events are triggered and processed according to the authentication events policy.
7. The application reacts to events from Azure Event Grid and performs policy actions based on those events.
8. The application must include the Event Grid Event ID field in all Application Insights telemetry
9. space

Why Do I:

1. Azure Service Bus
    a. Order Processing
    b. the order of messages is important as it represents the flow of data
    c. require transactions, ordering, duplicate detection, and instantaneous consistency.
    d. publish alarms:
    e. Each alarm controller uses Azure Service Bus to receive alarm signals as part of a transaction. Alarm events must be recorded for audit purposes. Each transaction record must include information about the alarm type that was activated. You need to implement a reply trail auditing solution.
    f. Azure Service Bus messages, payloads, and serialization - Azure Service Bus | Microsoft Learn
    g. Azure Service Bus topic filters - Azure Service Bus | Microsoft Learn
        i. SQL filters

AZ Service Bus
SQL filter - C#

```
adminClient = new ServiceBusAdministrationClient(connectionString);

// Create a SQL filter with color set to blue and quantity to 10
await adminClient.CreateSubscriptionAsync(
        new CreateSubscriptionOptions(topicName, "ColorBlueSize10Orders"),
        new CreateRuleOptions("BlueSize10Orders", new SqlRuleFilter("color='blue' AND quantity=10"
        )));

// Create a SQL filter with color set to red
// Action is defined to set the quantity to half if the color is red
await adminClient.CreateRuleAsync(topicName, "ColorRed", new CreateRuleOptions
{
    Name = "RedOrdersWithAction",
    Filter = new SqlRuleFilter("user.color='red'"),
    Action = new SqlRuleAction("SET quantity = quantity / 2;")
}
```

ii.

1. tests for the existence of properties (EXISTS), null-values (IS NULL), logical NOT/AND/OR, relational operators, simple numeric arithmetic, and simple text pattern matching with LIKE.

iii. Boolean filters

```
// Create a True Rule filter with an expression that always evaluates to true
// It's equivalent to using SQL rule filter with 1=1 as the expression
await adminClient.CreateSubscriptionAsync(
        new CreateSubscriptionOptions(topicName, subscriptionAllOrders),
        new CreateRuleOptions("AllOrders", new TrueRuleFilter()));
```

Service Bus - C# Boolean filter

iv.

v. Correlation filters

vi. A set of conditions that are matched against one or more of an arriving *message's user and system properties*.

1. ContentType
2. Label
3. MessageId
4. ReplyTo
5. ReplyToSessionId
6. SessionId
7. To
8. any user-defined properties.

vii. A match exists when an arriving message's value for a property is equal to the value specified in the correlation filter.

viii. For string expressions, the comparison is case-sensitive.

ix. If you specify multiple match properties, the filter combines them as a logical AND condition, meaning for the filter to match, all conditions must match.

```
// Create a correlation filter with color set to Red and priority set to High
await adminClient.CreateSubscriptionAsync(
        new CreateSubscriptionOptions(topicName, "HighPriorityRedOrders"),
        new CreateRuleOptions("HighPriorityRedOrdersRule", new
        CorrelationRuleFilter() {Subject = "red", CorrelationId = "high"} ));
```

AZ Service Bus - C# - Corellation filter

h.

i. space

2. Queue FIFO storage
    a. asynchronously communicate transaction information by using REST messages
3. Grid
    a. Respond to events while processing an order
        i. Users must be informed as and when there is progress for their order.
    b. Pub – sub (topics)

   c. ValidationCode Handshake: requires you to prove ownership of your Webhook endpoint before it starts delivering events to that endpoint.
   d. Ensure that authentication events are triggered and processed according to the authentication events policy.
   e. Trigger the workflow whenever a deallocation activity occurs for a virtual machine.
4. Hubs
   a. Analyze live user behavior
   b. 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.
   c. capture, retention, and replay of telemetry and event stream data.
5. You are developing an e-commerce application.
6. The application generates millions of orders and captures user behavior.
7. Users must be informed as and when there is progress for their order.
8. The application must support sales and marketing team to analyze live user behavior.
9. Which Azure service should you use respond to events while processing an order?
10. Which Azure service should you use to support order processing?
11. Which Azure service should you use to support sales and marketing team?

Consistency Levels – Cosmos DB

1. Consistency levels in Azure Cosmos DB | Microsoft Learn

2.
3. Bounded Staleness reasons:
   a. Latency must be low for data write operations and require order guarantee between distributed databases.
4. Strong
   a. No out of order, all is current
5. Bounded Staleness
   a. latency must be low for data write operations and require order guarantee between distributed databases.
6. Eventual
   a. The application can afford to read out of order writes.
7. Consistent Prefix
8. `cosmosdb create`
9. `az copy`

azcopy cp " Slot1 "
" Slot2 /? ?sv=2018-03-28&ss=bjqt&srt=sco&sp=rwddgcup&se=2019-05-01T05:01:17Z&st=2019-04-30T21:01:17Z&spr=https&sig=MGCXiyEzbtttkr3ewJIh2AR8KrghSy1DGM9ovN734bQF4%3D"
Slot3

10.

Option 2: Use a SAS token

You can append a SAS token to each source or destination URL that use in your AzCopy commands.

This example command recursively copies data from a local directory to a blob container. A fictitious SAS token is appended to the end of the of the container URL.

AzCopy Copy

```
9-05-01T05:01:17Z&st=2019-04-30T21:01:17Z&spr=https&sig=MGCXiyEzbtttkr3ewJIh2AR8KrghSy1DGM9ovN734bQF4%3D" --recursive=true
```

11.
12. Consistency level vs. Consistency in code:

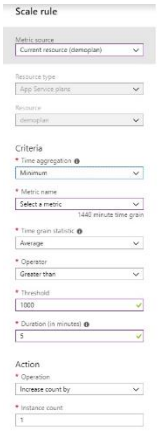
Actions

1. Event
    a. a user logs on, the web service notifies all the client applications so they can display that user's status as "Online"
    b. The login notification is an event: it contains only a simple piece of status data and there is no expectation by the authentication service for the client applications to react to the notice in any particular way.
2. Message
    a. The delete-account notification is a message
3. Service Bus Queue trigger.
    a. add a message to an Azure Service Bus Queue from the mobile application.
        i. Create an Azure Function App that uses an Azure Service Bus Queue trigger.
    b. QueueTrigger lets you trigger the Function when a message arrives in the queue.
4. Ephemeral OS disks are created on the local virtual machine (VM) storage and not saved to the remote Azure Storage. Ephemeral OS disks work well for stateless workloads
5. The QueueTrigger lets you trigger the Function when a message arrives in the queue.
6.
7. The Azure Cosmos DB output binding lets you write a new document to an Azure Cosmos DB database using the SQL API.
8. The direction must be set to out.
9. Azure Cosmos DB output binding for Functions 2.x and higher | Microsoft Learn

Application Insights resource

10. Create a URL ping test
11. Test frequency – Sets how often the test is run from each test location. With a default frequency of five minutes and five test locations, your site is tested on average every minute.
12. Monitor availability with URL ping tests - Azure Monitor | Microsoft Learn – 2022

Exam Notes



- ○ Storage queue
- ◉ Service Bus queue

They have defined a scale out condition that will scale the App if the average number of Active messages in a service bus queue is greater than 1000.

You need to ensure that the App service continually scales down with a scale down rule.

- ◉ Successful requests
- ○ Incoming requests
- ○ Active Messages Processed
- ○ Size

Metric
Metric source
Operator for the rule
Operation for the rule

Which of the following would you choose as the metric source?

- ○ Current resource(demoplan)
- ○ Storage queue
- ◉ Service Bus queue
- ○ Application Insights

- ☑ Create a Service Bus topic for each restaurant for which a driver can receive messages.
- ☐ Create a single Service Bus topic
- ☐ Create a single Service Bus subscription
- ☑ Create a single Service Bus Namespace
- ☐ Create a Service Bus Namespace for each restaurant for which a driver can receive messages.
- ☑ Create a Service Bus Subscription for each restaurant for which a driver can receive messages.

make use of the Azure Service Bus service.

Uber Eats

restaurants

· first driver to accept an order removes it from the list of available orders.

AZ CLI

```
az Slot1 --name bhuvangroup --location eastus
az Slot2 \      create a virtual machine
  --resource-group bhuvangroup \
       --name bhuvanvm \
     Slot3  win2016datacenter \
       --admin-username bhuvanlabsr \
       --admin-password Bhuvanr123$23
```

Others to see

Create an Azure Service Bus instance by providing a name, pricing tier, subscription, resource group, and location.

Azure API Management

The API provides product data to external consultants.

– Support alternative input parameters.

– Remove formatting text from responses.

– Provide additional context to back-end services.

- ○ Inbound
- ○ Outbound
- ◉ Backend
- ○ Error

Forward the user ID that is associated with the subscription key for the original request to the back-end service

Docker file syntax

Run a

powershell script called appscript.ps1 in the Docker container

the DockerFile

- ☑ FROM microsoft/dotnet:2.2-aspnetcore-runtime
- ☐ EXPOSE myApp.dll ,appscript.ps1
- ☑ ENTRYPOINT ["dotnet", " myApp.dll "]
- ☐ ENTRYPOINT ["myApp.dll" ,"appscript.ps1"]
- ☑ RUN powershell "appscript.ps1"
- ☐ RUN "myApp.dll","appscript.ps1"

requirements is to ensure that if users make requests based on passing an ID parameter, then those requests should always be served from a Point of Presence. An example of the URL is given below

Azure CDN to route requests to the Web App

https://bhuvanlabs.com/Customer.aspx?ID=1

- ○ Ignore query strings
- ○ Default setting
- ○ Bypass caching
- ○ Cache every unique URL

PowerShell verb - noun New-AzureRmServiceBusQueue

New-AzResourceGroup - PowerShell

Starts with az noun verb

Azure CLI az group create (no - between)

Which Azure CLI or PowerShell

create an Azure Service Bus instance by providing a name, pricing tier, subscription, resource group, and location.

- ○ New-AzureRmServiceBusQueue -ResourceGroupName "bhuvanlabs-rg" -NamespaceName bhuvanlabs -Name bhuvanlabsqueue -EnablePartitioning $False
- ○ az group create --name "bhuvanlabs-rg" --location "Central US"
- ○ New-AzureRmResourceGroup -Name "bhuvanlabs-rg" -Location "Central US"
- ○ New-AzureRmServiceBusNamespace -ResourceGroup "bhuvanlabs-rg" -NamespaceName bhuvanlabs -Location WestUS -SkuName "Standard"

Azure Redis

```
class BhuvanlabEmployee
{
    public string Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }

    public Employee(string EmployeeId, string Name, int Age)
    {
        this.Id = EmployeeId;
        this.Name = Name;
        this.Age = Age;
    }
}
```

- SetObject
- SerializeObject
- GetObject
- SerializeClass

```
// Store object to cache
BhuvanlabEmployee obj = new BhuvanlabEmployee("007", "Davide Columbo", 100);
Console.WriteLine("Cache response from storing Employee .NET object : " +
cache. Slot1 ("e007", JsonConvert. Slot2 (obj)));

// Retrieve object from cache
BhuvanlabEmployee e007FromCache = JsonConvert. Slot3 <BhuvanlabEmployee>(cache. Slot4 ("e007"));
```

Slot2?

```
// Create a True Rule filter with an expression that always evaluates to true
// It's equivalent to using SQL rule filter with 1=1 as the expression
await adminClient.CreateSubscriptionAsync(
        new CreateSubscriptionOptions(topicName, subscriptionAllOrders),
        new CreateRuleOptions("AllOrders", new TrueRuleFilter()));
```

Service Bus - C# Boolean filter

```
// Create a correlation filter with color set to Red and priority set to High
await adminClient.CreateSubscriptionAsync(
        new CreateSubscriptionOptions(topicName, "HighPriorityRedOrders"),
        new CreateRuleOptions("HighPriorityRedOrdersRule", new
        CorrelationRuleFilter() {Subject = "red", CorrelationId = "high"} ));
```

AZ Service Bus - C# - Corellation filter

```
adminClient = new ServiceBusAdministrationClient(connectionString);

// Create a SQL filter with color set to blue and quantity to 10
await adminClient.CreateSubscriptionAsync(
        new CreateSubscriptionOptions(topicName, "ColorBlueSize10Orders"),
        new CreateRuleOptions("BlueSize10Orders", new SqlRuleFilter("color='blue' AND quantity=10"
        )));

// Create a SQL filter with color set to red
// Action is defined to set the quantity to half if the color is red
await adminClient.CreateRuleAsync(topicName, "ColorRed", new CreateRuleOptions
{
    Name = "RedOrdersWithAction",
    Filter = new SqlRuleFilter("user.color='red'"),
    Action = new SqlRuleAction("SET quantity = quantity / 2;")
}
```

AZ Service Bus
SQL filter - C#

Push notifications

implement push notifications

```csharp
static void ReceiveMessageAndSendNotification(string connectionString)
{
    // Initialize the Notification Hub
    string hubConnectionString = CloudConfigurationManager.GetSetting
            ("Microsoft.NotificationHub.ConnectionString");
    hub = NotificationHubClient.CreateClientFromConnectionString
            (hubConnectionString, "enterprisepushservicehub");

    SubscriptionClient Client =
        SubscriptionClient.CreateFromConnectionString
                (connectionString, sampleTopic, sampleSubscription);

    Client.Receive();

    // Continuously process messages received from the subscription
    while (true)
    {
        BrokeredMessage message = Client.Receive();
        var toastMessage = @"<toast><visual><binding template=""ToastText01""><text id=""1"">{messag

        if (message != null)
        {
            try
            {
                Console.WriteLine(message.MessageId);
                Console.WriteLine(message.SequenceNumber);
                string messageBody = message.GetBody<string>();
                Console.WriteLine("Body: " + messageBody + "\n");

                toastMessage = toastMessage.Replace("{messagepayload}", messageBody);
                SendNotificationAsync(toastMessage);
```

Trick question – what is not mentioned is already created

A company is developing a solution that allows smart devices to send information to a central location. The solution must receive and store messages until they can be processed. You create an Azure Service Bus instance by providing a name, pricing tier, subscription, resource group, and location.

Since the question already states that we have a resource group and namespace in place, we can just use the New-AzureRmServiceBusQueue to create a new queue in the namespace

Which Azure CLI or PowerShell command should you run?

- New-AzureRmServiceBusQueue -ResourceGroupName "bhuvanlabs-rg" -NamespaceName bhuvanlabs -Name bhuvanlabsqueue -EnablePartitioning $False
- az group create --name "bhuvanlabs-rg" --location "Central US"
- New-AzureRmResourceGroup -Name "bhuvanlabs-rg" -Location "Central US"
- New-AzureRmServiceBusNamespace -ResourceGroup "bhuvanlabs-rg" -NamespaceName bhuvanlabs -Location WestUS -SkuName "Standard"

Add the following `Employee` class definition to *Program.cs*:

C# Copy

```
class Employee
{
    public string Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }

    public Employee(string EmployeeId, string Name, int Age)
    {
        this.Id = EmployeeId;
        this.Name = Name;
        this.Age = Age;
    }
}
```

At the bottom of `Main()` procedure in *Program.cs*, and before the call to `Dispose()`, add the following lines of code to cache and retrieve a serialized .NET object:

C# Copy

```
// Store .NET object to cache
Employee e007 = new Employee("007", "Davide Columbo", 100);
Console.WriteLine("Cache response from storing Employee .NET object : " +
    cache.StringSet("e007", JsonConvert.SerializeObject(e007)));

// Retrieve .NET object from cache
Employee e007FromCache = JsonConvert.DeserializeObject<Employee>(cache.StringGet("e007"));
Console.WriteLine("Deserialized Employee .NET object :\n");
Console.WriteLine("\tEmployee.Name : " + e007FromCache.Name);
Console.WriteLine("\tEmployee.Id   : " + e007FromCache.Id);
Console.WriteLine("\tEmployee.Age  : " + e007FromCache.Age + "\n");
```

The next query performs iteration over `children` in the `Families` container. The output array is different from the preceding query. This example splits `children`, and flattens the results into a single array:

SQL

```
SELECT *
FROM c IN Families.children
```

The results are:

JSON

```
[
  {
      "firstName": "Henriette
      "gender": "female",
      "grade": 5,
      "pets": [{ "givenName":
  },
  {
      "familyName": "Merriam",
      "givenName": "Jesse",
      "gender": "female",
      "grade": 1
  },
  {
      "familyName": "Miller",
      "givenName": "Lisa",
      "gender": "female",
      "grade": 8
  }
]
```

Upload blobs to a container

The following code snippet gets a reference to a **CloudBlockBlob** object by calling the GetBlockBlobReference method on the container created in the previous section. It then uploads the selected local file to the blob by calling the UploadFromFileAsync method. This method creates the blob if it doesn't already exist, and overwrites it if it does.

C# Copy

```
// Create a file in your local MyDocuments folder to upload to a blob.
string localPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string localFileName = "QuickStart_" + Guid.NewGuid().ToString() + ".txt";
string sourceFile = Path.Combine(localPath, localFileName);
// Write text to the file.
File.WriteAllText(sourceFile, "Hello, World!");

Console.WriteLine("Temp file = {0}", sourceFile);
Console.WriteLine("Uploading to Blob storage as blob '{0}'", localFileName);

// Get a reference to the blob address, then upload the file to the blob.
// Use the value of localFileName for the blob name.
CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(localFileName);
await cloudBlockBlob.UploadFromFileAsync(sourceFile);
```

8. Question

View CaseStudy: https://rb.gy/zuy49h

You need to formulate a query that would be used to get all items where the order price is $9.99

Select * from c IN **Slot1** where **Slot2** = "9.99"

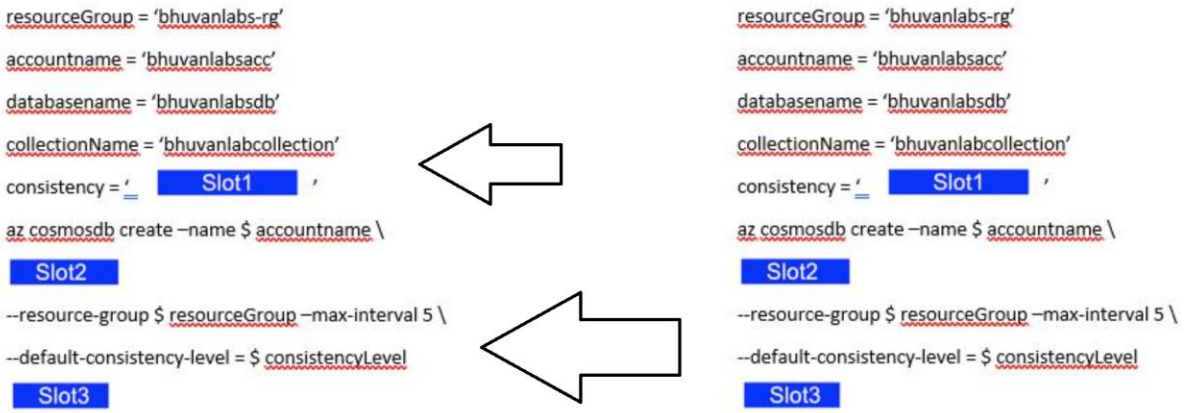
Which of the following would go into Slot1?

- orders
- customer
- course
- customer.orders

Incorrect

Answer D

Azure CLI Samples for Azure Cosmos DB | Microsoft Learn



Which of the following would go into Slot1?

- Strong
- Eventual
- ConsistentPrefix
- BoundedStaleness

Which of the following would go into Slot3?

- –locations 'southecentralus'
- –locations 'eastus'
- --locations southcentalus=0 eastus=1
- --locations southecentralus=0

Request

The `Lease Blob` request may be constructed as follows. HTTPS is recommended. Replace *myaccount* with the name of your storage account:

PUT Method Request URI	HTTP Version
`https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=lease`	HTTP/1.1

YAML

```yaml
apiVersion: '2018-10-01'
location: eastus
name: secret-volume-demo
properties:
  containers:
  - name: aci-tutorial-app
    properties:
      environmentVariables: []
      image: mcr.microsoft.com/azuredocs/aci-helloworld:latest
      ports: []
      resources:
        requests:
          cpu: 1.0
          memoryInGB: 1.5
      volumeMounts:
      - mountPath: /mnt/secrets
        name: secretvolume1
  osType: Linux
  restartPolicy: Always
  volumes:
  - name: secretvolume1
    secret:
      mysecret1: TXkgZmlyc3Qgc2VjcmV0IEZPTwo=
      mysecret2: TXkgc2Vjb25kIHNlY3JldCBCQVIK
tags: {}
type: Microsoft.ContainerInstance/containerGroups
```

Yaml name value pairs goto new line

Subscribe to a custom topic

You subscribe to an event grid topic to tell Event Grid which events you want to track. The following example subscribes to the custom topic you created, and passes the resource ID of the event hub for the endpoint. The endpoint is in the format:

`/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.EventHub/namespaces/<namespace-name>/eventhubs/<hub-name>`

The following script gets the resource ID for the event hub, and subscribes to an event grid topic. It sets the endpoint type to `eventhub` and uses the event hub ID for the endpoint.

Azure CLI | Copy | Try It

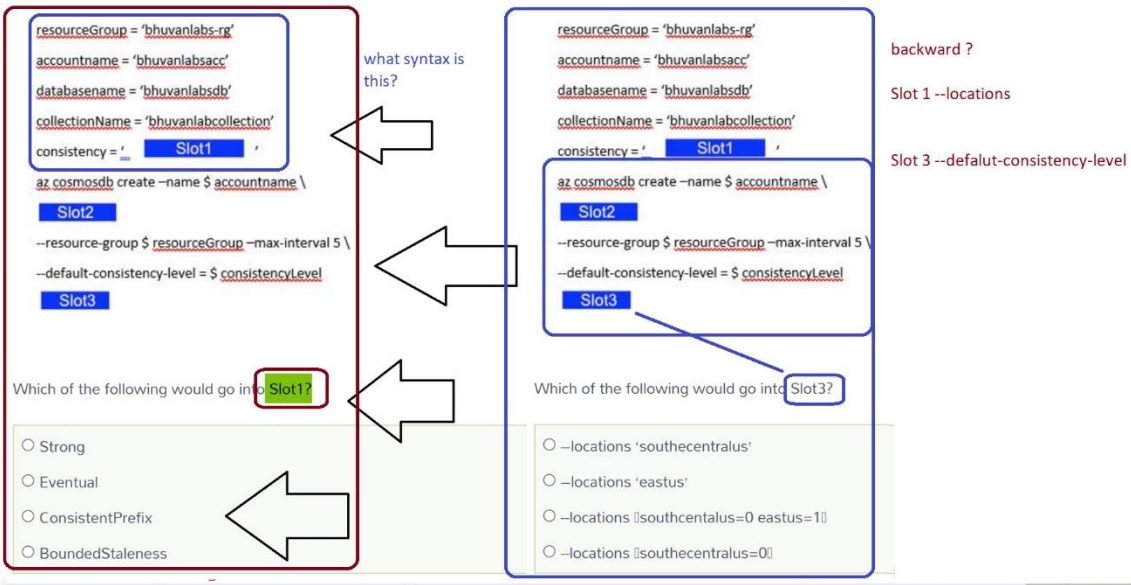
```
hubid=$(az eventhubs eventhub show --name $hubname --namespace-name $namespace --re

az eventgrid event-subscription create \
  --topic-name $topicname \
  -g gridResourceGroup \
  --name subtoeventhub \
  --endpoint-type eventhub \
  --endpoint $hubid
```

The account that creates the event subscription must have write access to the event hub.

application which would be used to interact with Azure Event Hubs. You have created an Azure Event Hub namespace and the Event Hub itself

C# Copy

```
static async Task Main()
{
    // Create a producer client that you can use to send events to an event hub
    await using (var producerClient = new EventHubProducerClient(connectionStrir
    {
        // Create a batch of events
        using EventDataBatch eventBatch = await producerClient.CreateBatchAsync(

        // Add events to the batch. An event is a represented by a collection of
        eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("First event")));
        eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("Second event")))
        eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("Third event")));

        // Use the producer client to send the batch of events to the event hub
        await producerClient.SendAsync(eventBatch);
        Console.WriteLine("A batch of 3 events has been published.");
    }
}
```




```
az cosmosdb create --name $account --resource-group $resourceGroup --default-
consistency-level Eventual --locations regionName="$location" failoverPriority=
0 isZoneRedundant=False --locations regionName="$failoverLocation"
failoverPriority=1 isZoneRedundant=False
```

CRON Job Format: ****** - min – hour – day – month – dow

implement the bindings for the CheckUserContent function.
How should you complete the code segment?

```
public static class CheckUserContent
{
    [FunctionName ("CheckUserContent")]
    public static void Run(
        [QueueTrigger("userContent")] / [BlobTrigger("userContent/{name}")] / [CosmosDBTrigger("content", "userContent")] / [Table("content", "userContent", "{name}")]
        string content,
        [Queue("userContent")] / [CosmosDB("content", "userContent")] / [Table("content", "userContent", "{name}")] / [Blob("userContent/{name}", FileAccess.Write)]
        Stream output)
    {
        ...
    }
}
```

```
var resolver = new KeyVaultKeyResolver(_keyVaultClient);
var keyBundle = await _keyVaultClient.GetKeyAsync("...", "...");
```

<Code Snippet1>

<Code Snippet2>

<Code Snippet3>

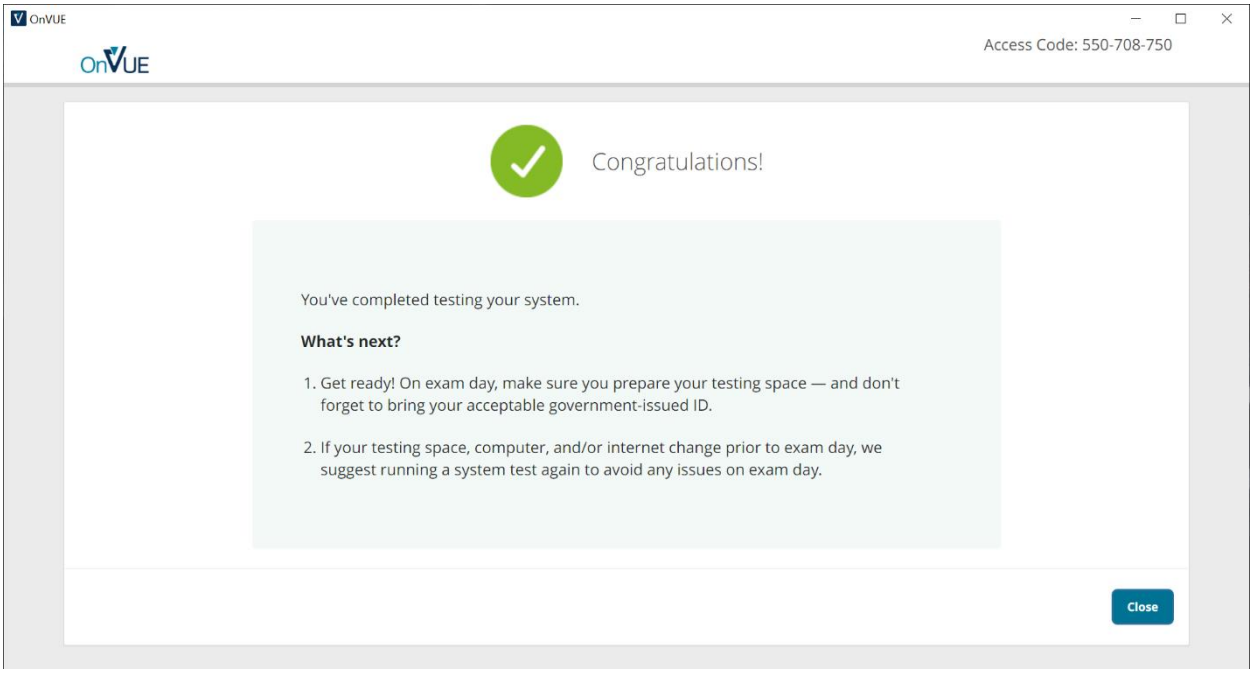
```
1  //Encrypt a blob and upload.
2
3  // Retrieve the key that you created previously.
4  // The IKey that is returned here is an RsaKey.
5  var rsa = cloudResolver.ResolveKeyAsync(
6  "https://contosokeyvault.vault.azure.net/keys/TestRSAKey1",
7  CancellationToken.None).GetAwaiter().GetResult();
8
9  // Now you simply use the RSA key to encrypt by setting it in the
   BlobEncryptionPolicy.
10 BlobEncryptionPolicy policy = new BlobEncryptionPolicy(rsa, null);
11 BlobRequestOptions options = new BlobRequestOptions() { EncryptionPolicy
   = policy };
12
13 // Reference a block blob.
14 CloudBlockBlob blob = contain.GetBlockBlobReference("MyFile.txt");
15
16 // Upload using the UploadFromStream method.
17 using (var stream = System.IO.File.OpenRead(@"C:\Temp\MyFile.txt"))
18 blob.UploadFromStream(stream, stream.Length, null, options, null);
```

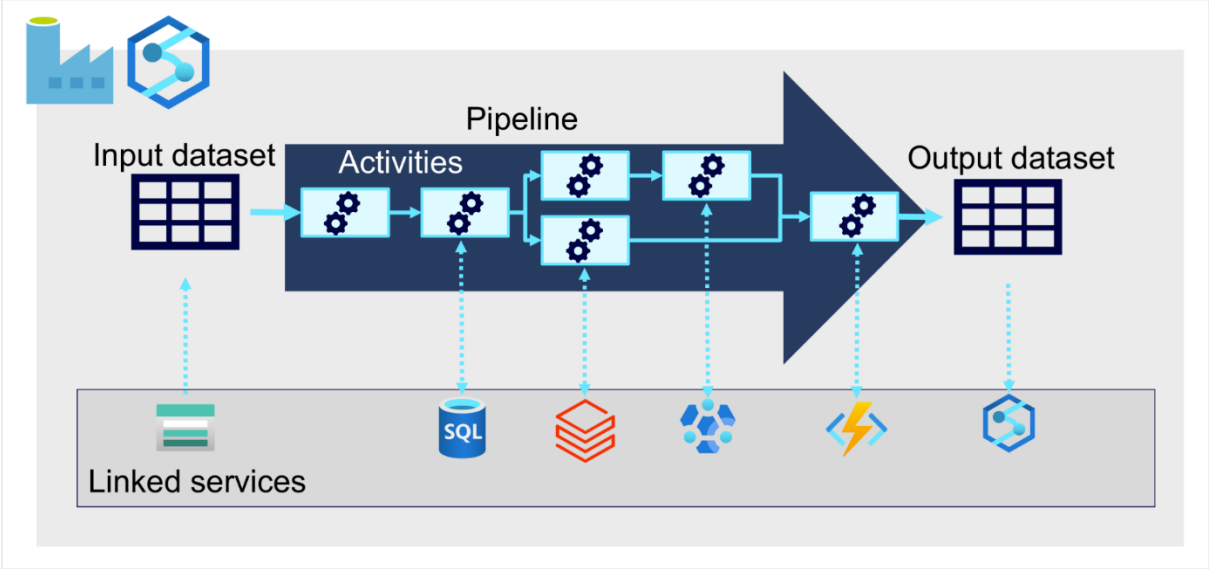
```
CS01 $storageAccount = Get-AzureRmStorageAccount -ResourceGroupName "..." -AccountName "...
CS02 $keyVault = Get-AzureRmKeyVault -VaultName "..."
CS03 $key = Get-AzureKeyVaultKey -VaultName $keyVault.VaultName -Name "..."
CS04  Set-AzureRmKeyVaultAccessPolicy `
CS05   -VaultName $keyVault.VaultName `
CS06   -ObjectId $storageAccount.Identity.PrincipalId `
CS07    PermissionsToKeys wrapkey, unwrapkey, get
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 the security policies are met.

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




Pipeline
Input dataset
Activities
Output dataset
SQL
Linked services