Motjoka Fanana ST10089515 CLDV6212 POE PART 3

A.

|  |  |  |
| --- | --- | --- |
| **COMPONENT** | **TECHNOLOGY CHOICE** | **HOSTING MODEL** |
| AZURE FUNCTION | COMPUTE | SERVERLESS |
| AZURE SQL DATABASE | DATA STORAGE | PaaS |
| AZURE QUEUE STORAGE | DATA STORAGE | PaaS |
| AZURE HTTP TRIGGER | COMPUTE | SERVERLESS |

B.

Motivation for Change: Cost Optimization, Scalability, and Isolation

1. Efficient Data Organization:

- Azure Table Storage, as a NoSQL data store, excels in organizing structured data with a key/attribute format. Shifting from a Storage Queue Trigger to Azure Table Storage allows for a more organized and query-friendly approach to storing and retrieving data.

2. Seamless Integration with Existing Systems:

- Azure Table Storage seamlessly integrates with other Azure services. By transitioning from Storage Queue Trigger to Azure Table Storage, you ensure compatibility and a smooth interaction with the broader Azure ecosystem. This integration fosters a cohesive and interconnected architecture.

3. Cost-Effective Data Handling:

- Cost optimization is a critical factor in any system. Azure Table Storage offers a cost-effective solution, especially when compared to the potentially higher costs associated with managing a large number of messages in storage queues. By leveraging Azure Table Storage, you can optimize costs without compromising on performance.

4. Streamlined Query Capabilities:

- Azure Table Storage provides powerful query capabilities, allowing you to retrieve specific data efficiently. This is particularly beneficial when dealing with diverse data associated with different vaccination records. The ability to query and filter data directly from Azure Table Storage enhances the speed and accuracy of information retrieval.

5. Enhanced Scalability:

- Azure Table Storage is designed to scale horizontally, accommodating increasing data volumes and request rates. This scalability is crucial in scenarios where the demand for vaccination-related data may grow rapidly. Transitioning to Azure Table Storage ensures that your system can seamlessly handle increased workloads as needed.

6. Improved Data Relationships:

- Unlike simple queue-triggered systems, Azure Table Storage enables you to establish relationships between different pieces of data. This relational structure can enhance the representation of complex associations within vaccination records, providing a more comprehensive view of the data.

7. Fine-Grained Access Control:

- Azure Table Storage offers fine-grained access control mechanisms, allowing you to define specific permissions for different operations. This enhances security and ensures that only authorized entities can perform certain actions on the stored data, contributing to a more robust and secure system.

In summary, the decision to transition from a Storage Queue Trigger to Azure Table Storage is driven by the desire for cost optimization, enhanced scalability, streamlined query capabilities, improved data relationships, and seamless integration within the broader Azure ecosystem. Embracing Azure Table Storage aligns with the evolving needs of efficiently managing and retrieving vaccination-related data in a scalable and cost-effective manner.

C. D.

CODE

using System;

using System.Data.SqlClient;

using Microsoft.Azure.WebJobs;

using Microsoft.Extensions.Logging;

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Queue;

using Microsoft.WindowsAzure.Storage.Table;

using Serilog;

public class Function1

{

[FunctionName("Function1")]

public void Run([QueueTrigger("message-queue", Connection = "MyQueueCon")] string myQueueItem, Microsoft.Extensions.Logging.ILogger log)

{

string Connstri = "Server=tcp:queuestorageserver1.database.windows.net,1433;Initial Catalog=dbQueues;Persist Security Info=False;User ID=kops;Password=Dbzgt1103;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;";

try

{

string[] attributes = myQueueItem.Split(':');

//INSERT VALUES HERE

string firstName = "Kopano";

string lastName = "Fanana";

string id = "02100151090";

string center = "WATERFALL";

string vaccinationDate = "2023/05/06";

string serialNumber = "RRT7752";

log.LogInformation($"Processing queue ID: {myQueueItem}");

// Add the message to the 'message-queue'

string queueName = "message-queue";

CloudStorageAccount cloudStorageAccount = CloudStorageAccount.Parse("DefaultEndpointsProtocol=https;AccountName=sakopano;AccountKey=0uJk9x/GShmzpqNjEigFSH5NXps5BMkAv13XTlreMmAV9IS/q9l97jHBU2ig7kPmdCm9OekazXYX+AStfC5QXw==;EndpointSuffix=core.windows.net");

CloudQueueClient queueClient = cloudStorageAccount.CreateCloudQueueClient();

CloudQueue queue = queueClient.GetQueueReference(queueName);

string queueMessageContent = $"{firstName}:{lastName}:{id}:{center}:{vaccinationDate}:{serialNumber}";

CloudQueueMessage queueMessage = new CloudQueueMessage(queueMessageContent);

queue.AddMessageAsync(queueMessage);

// Log successful message addition to the queue

log.LogInformation($"Queue Message Added To the 'message-queue' successfully, Id = {id}");

using (SqlConnection connection = new SqlConnection(Connstri))

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

// SQL command to insert data into the 'messages\_final' table

command.CommandText = "INSERT INTO VacTable (firstName, lastName, Id, CENTER, VaccinationDate, SerialNumber) VALUES (@firstName, @lastName, @Id, @Center, @VaccinationDate, @SerialNumber)";

command.Parameters.AddWithValue("@firstName", firstName);

command.Parameters.AddWithValue("@lastName", lastName);

command.Parameters.AddWithValue("@Id", id);

command.Parameters.AddWithValue("@Center", center);

command.Parameters.AddWithValue("@VaccinationDate", vaccinationDate);

command.Parameters.AddWithValue("@SerialNumber", serialNumber);

command.ExecuteNonQuery();

// Add the entry to the 'VacTable'

CloudTableClient cloudTableClient = cloudStorageAccount.CreateCloudTableClient();

CloudTable VacTable = cloudTableClient.GetTableReference("VacTable");

VacTableEntity vaccinationEntry = new VacTableEntity(firstName, lastName, id, center, vaccinationDate, serialNumber);

TableOperation insertOp = TableOperation.Insert(vaccinationEntry);

VacTable.ExecuteAsync(insertOp);

}

}

}

catch (Exception ex)

{

log.LogError($"Error processing queue message: {ex.Message}");

}

}

public class VacTableEntity : TableEntity

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string Id { get; set; }

public string Center { get; set; }

public string VaccinationDate { get; set; }

public string SerialNumber { get; set; }

public VacTableEntity(string firstName, string lastName, string id, string center, string vaccinationDate, string serialNumber)

{

this.PartitionKey = id;

this.RowKey = center;

this.FirstName = firstName;

this.LastName = lastName;

this.Id = id;

this.Center = center;

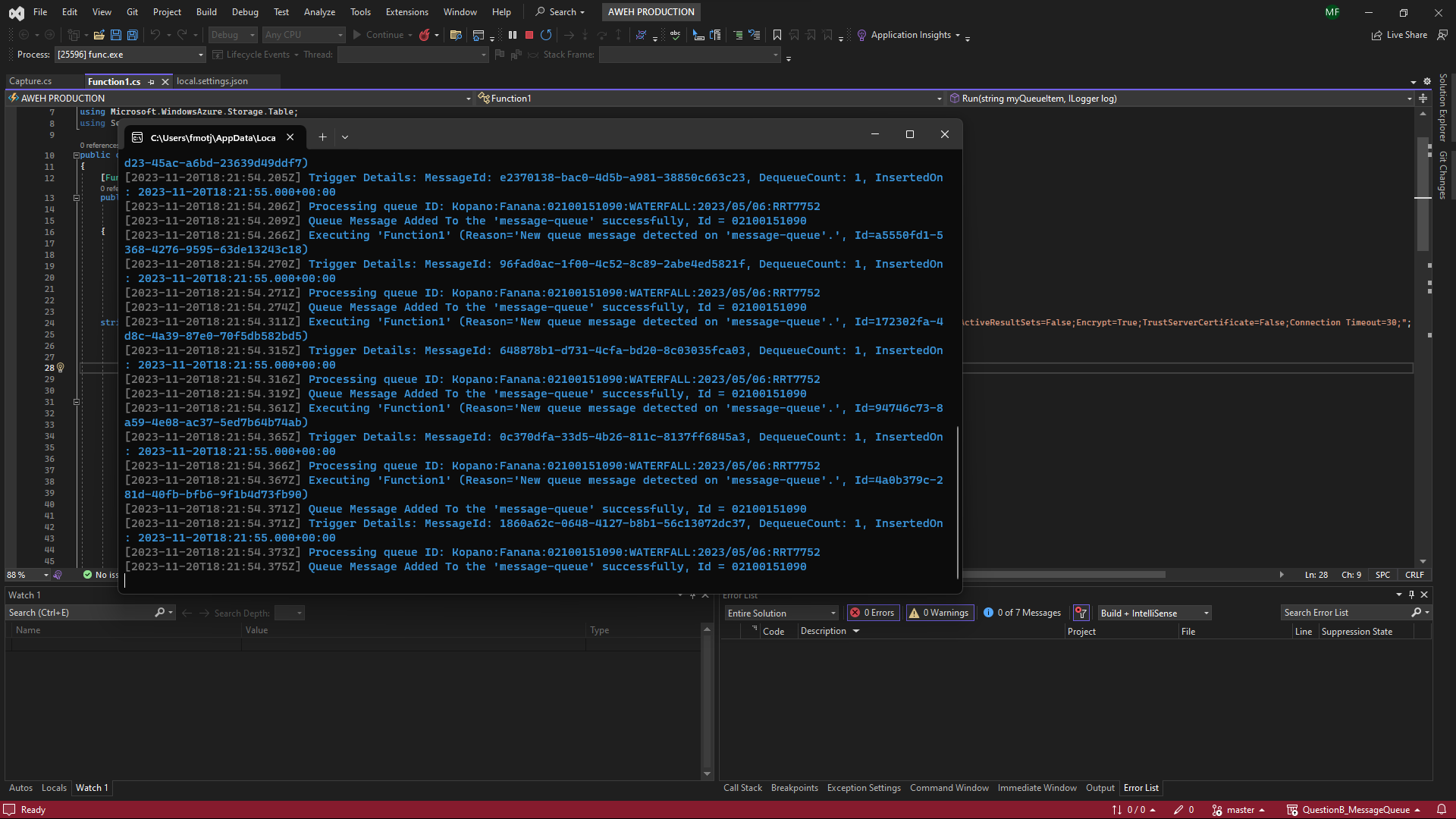
this.VaccinationDate = vaccinationDate;

this.SerialNumber = serialNumber;

}

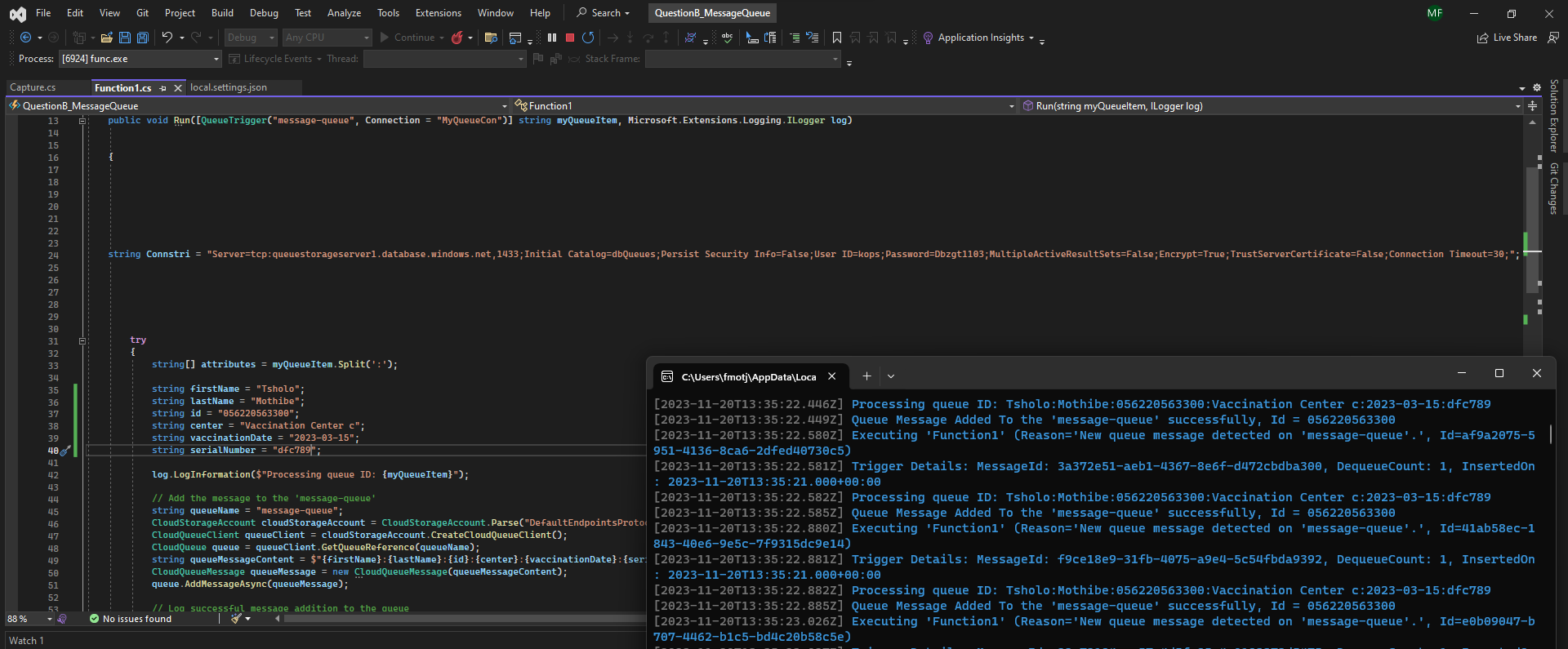
}

}



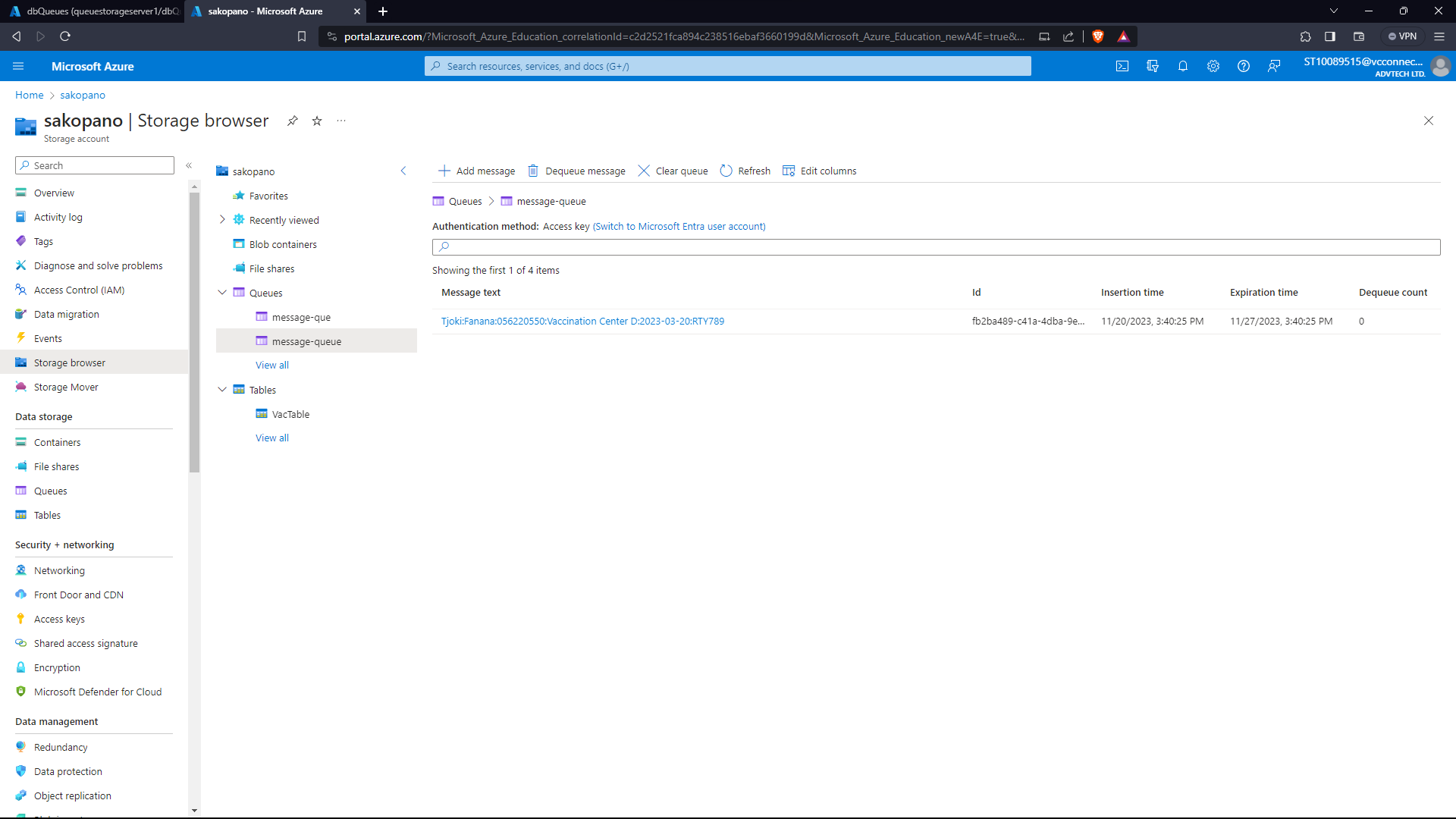
A screenshot of a computer

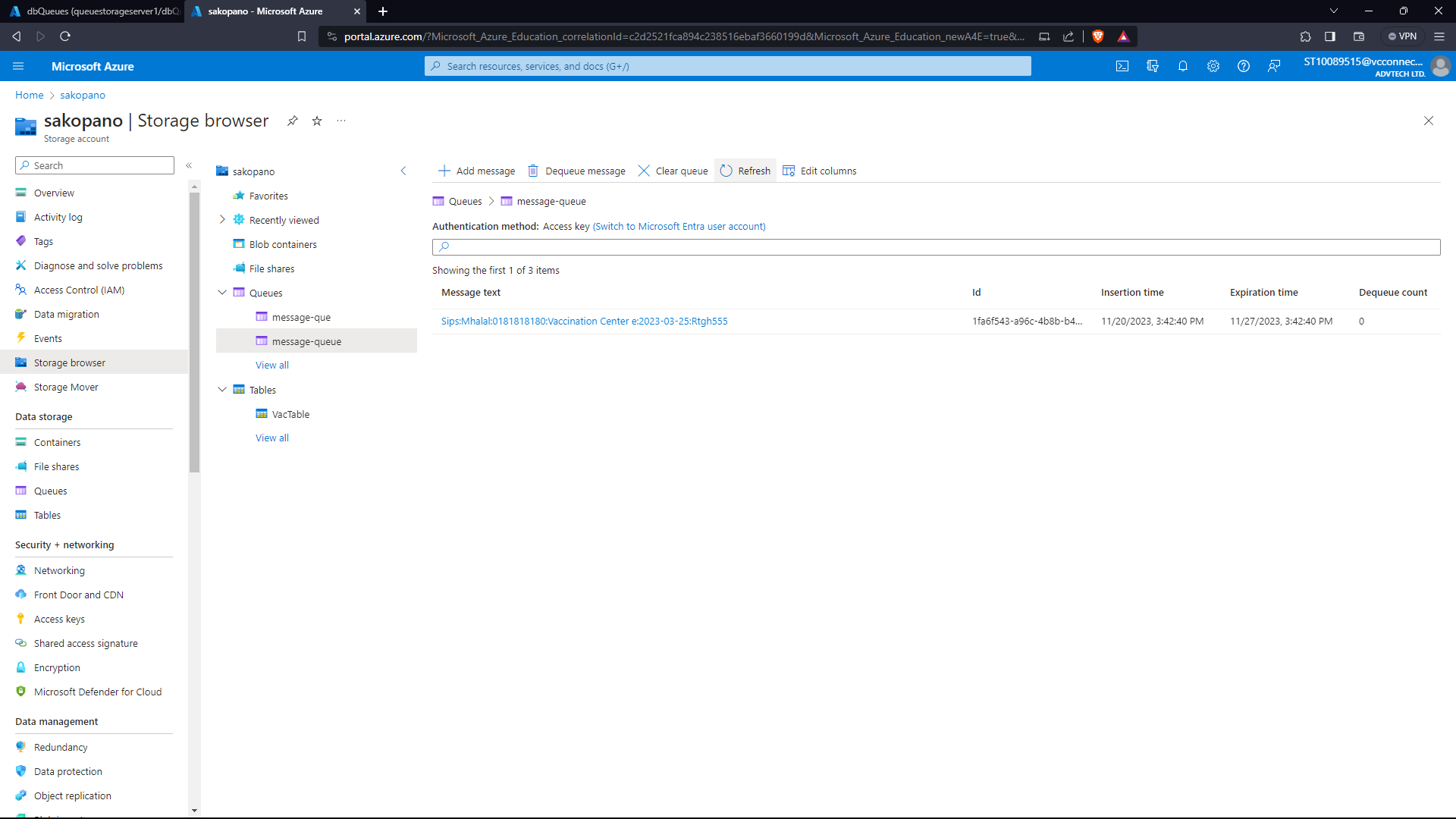
Description automatically generated

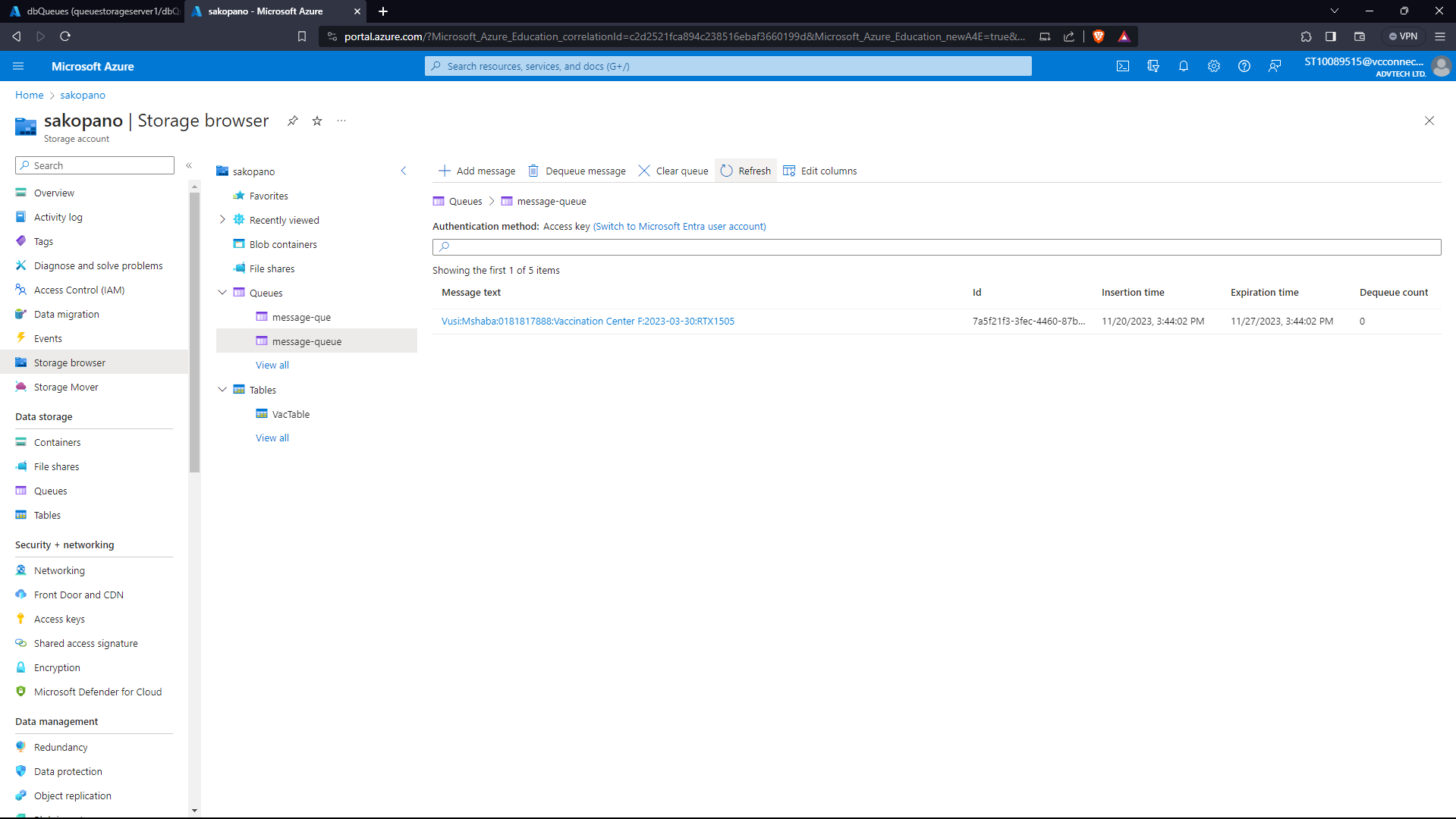


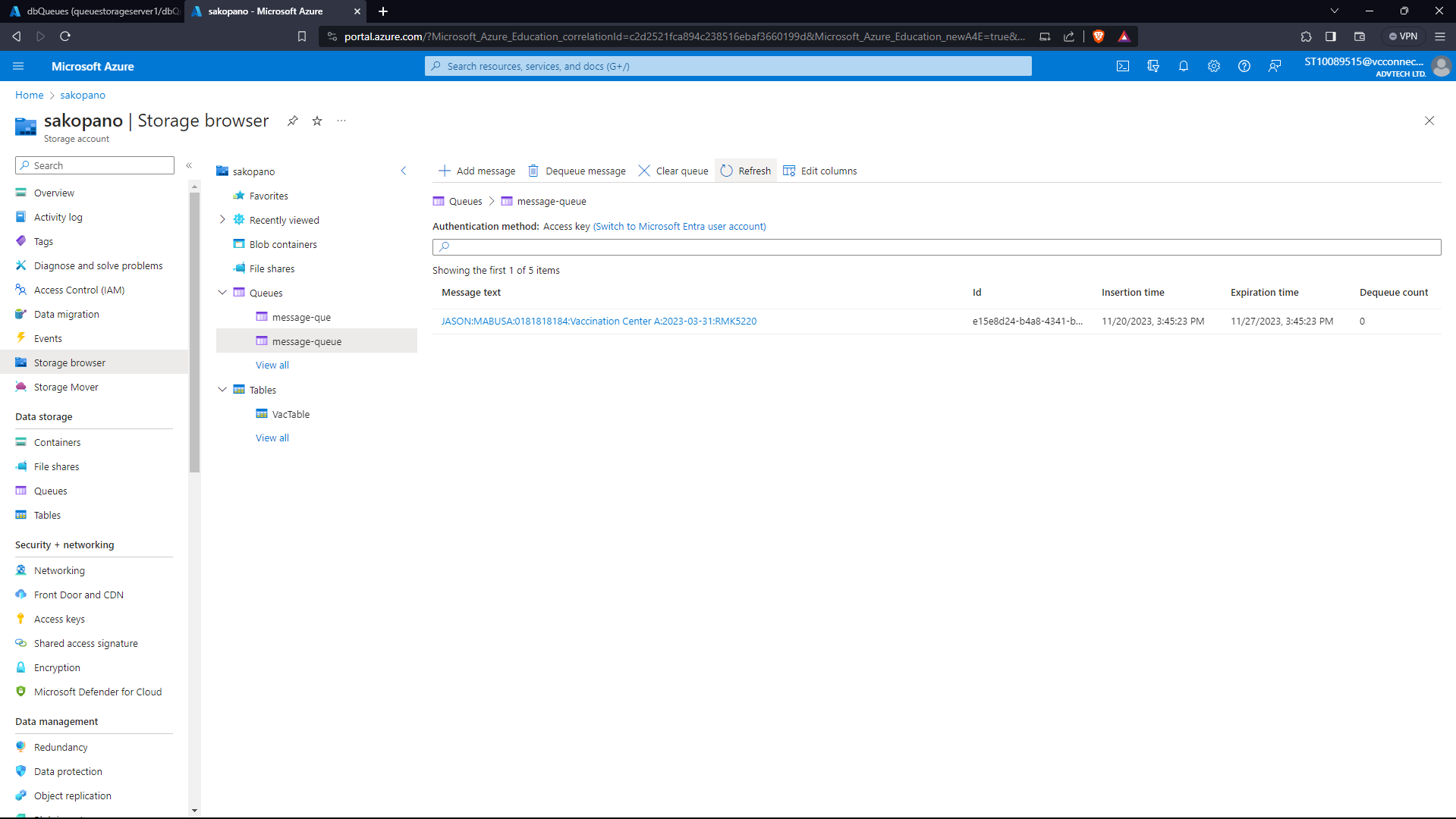
A screenshot of a computer

Description automatically generated









A screenshot of a computer

Description automatically generated

REFERENCES:

Microsoft (2023) *Azure blob storage*, *Microsoft Azure*. Available at: https://azure.microsoft.com/en-gb/products/storage/blobs/?ef\_id=\_k\_Cj0KCQiAmNeqBhD4ARIsADsYfTdkxQVhOgelHx--2J99oeR6iUjJf1L\_uX4DBZc4kK\_qx57fAXtnt0oaAimwEALw\_wcB\_k\_&OCID=AIDcmmck5pq7og\_SEM\_\_k\_Cj0KCQiAmNeqBhD4ARIsADsYfTdkxQVhOgelHx--2J99oeR6iUjJf1L\_uX4DBZc4kK\_qx57fAXtnt0oaAimwEALw\_wcB\_k\_&gad\_source=1 (Accessed: 16 November 2023).

Amazon, n.d. What is an Event-Driven Architecture?. [Online] .Available at: https://aws.amazon.com/event-driven-architecture/#:~:text=An%20event%2Ddriven%20architecture%20uses,on%20an%20e%2Dcommerce%20website. [Accessed 2 September 2023].

cassandrad, 2017 . Data Driven vs Event Driven model/architecture?. [Online]. Available at: https://stackoverflow.com/questions/42174856/data-driven-vs-event-driven-model-architecture .[Accessed 1 September 2023].

Fonzi, B., 2023. Choosing between Data and Event-Driven Architecture. [Online]. Available at: https://www.linkedin.com/pulse/choosing-between-data-event-driven-architecture-bruno-fonzi/ .[Accessed 3 September 2023].

Sommerville, I., 2016. Software Engineering. 10th ed. Boston: Pearson Education.

Vaughan, J., n.d. data. [Online]. Available at: https://www.techtarget.com/searchdatamanagement/definition/data#:~:text=In%20computing%2C%20data%20is%20information,subject%20or%20a%20plural%20subject. [Accessed 2 September 2023].

Microsoft. (2022). Azure Virtual Machines documentation. Microsoft Azure. https://docs.microsoft.com/en-us/azure/virtual-machines/ [Accessed 2 September 2023].

Smith, J. (2021). Best Practices for Azure Blob Storage. Microsoft Azure. https://azure.microsoft.com/en-us/resources/whitepapers/ [Accessed 25 September 2023].

Microsoft. (2023). Azure SQL Database documentation. Microsoft Azure. https://docs.microsoft.com/en-us/azure/sql-database/ [Accessed 22 October 2023].

Microsoft. (2022). Azure Active Directory documentation. Microsoft Azure. https://docs.microsoft.com/en-us/azure/active-directory/ [Accessed 22 October 2023].

Microsoft. (2021). Azure Functions documentation. Microsoft Azure. https://docs.microsoft.com/en-us/azure/azure-functions/ [Accessed 27 October2023].