	Hope Foundation's **Finolex Academy of Management and Technology, Ratnagiri** **Information Technology Department**		
Subject name: Cloud Service Design Lab			Subject Code: ITL603
Class	TE IT	Semester – VI (CBCGS)	Academic year: 2018-19
Name of Student	**Kazi Jawwad A Rahim**	**QUIZ Score : 06/10**	
Roll No	**27**	Assignment/Experiment No.	04
Title: To understand Azure Service Bus Configuration / Rabbit MQ Configuration			

1.Course objectives applicable

COB3. To describe steps to perform on demand Application delivery.

COB6.To describe the functioning of Platform as a Service.

2. Course outcomes applicable:

CO2 -To define cloud computing and memorize diff. cloud service and deployment models

CO6-Use and Examine different cloud computing services

3. Learning Objectives:

1. To understand concept of message queue
2. To configure Message Queuing Server

4. Practical applications of the assignment/experiment: Message Queues are used by cloud based application to communicate with each other

5. Prerequisites:

1. Knowledge of Cloud Application Architecture
2. Internet Access
3. Knowledge of Azure cloud, Service Bus

6. Hardware Requirements:

1. Internet Access with Browser
2. Access to Microsoft Azure Cloud Subscription

7. Software Requirements:

Browser like Chrome, Internet Explorer Edge , RabbitMQ Server, Visual Studio 2017

8. Quiz Questions (if any): (Online Exam will be taken separately batchwise, attach the certificate/ Marks obtained)

1. What is Service Bus?
2. Which version of Visual Studio is used?
3. What is an Access Key?

9. Experiment/Assignment Evaluation:

Sr. No.	Parameters	Marks obtained	Out of
1	Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used -		6
2	Neatness/presentation		2
3	Punctuality		2
Date of performance (DOP)		**Total marks obtained**	**10**
Date of checking (DOC)		**Signature of teacher**	

11. Installation Steps / Performance Steps –

Introduction

The Azure Service Bus Queue is different than the Azure Storage Queues. In my last article, we have seen how to get started with Azure Service Bus by creating a namespace. This article tells you how to work with Azure Service Bus Queues with a sample console application.

Content

- Service Bus Queues
- Create Service Bus Queues in Azure portal
- Send and receive a message using service bus Queues

Pre-request

- Visual Studio 2017 update 3 or later
- Azure Subscription

Service Bus Queues

- Service Bus Queues provide a queueing mechanism
- Message will appear only once
- Message is processed using FIFO(First In First out) pattern
- Support transactions

The message lock can be renewed, It consists of few major parts,

1. Body – The body can be serialized object or a stream
2. Label – Simple text label
3. Time to Live – How long the message is stored in queue
4. Properties – Dictionary of properties that can be used by your specific consume

Create a service bus

Step 1

Log in to the Azure portal *(http://portal.azure.com),* using your Azure account.

Step 2

Go to the service bus namespace which is already created.

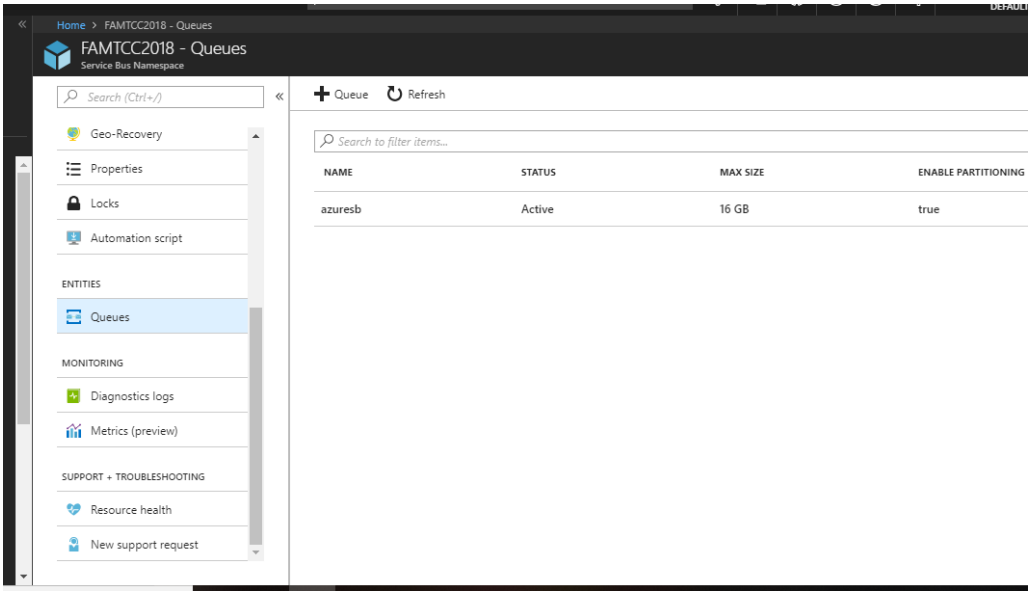

Figure 1: Service Bus Namespace

Step 3

Click on Queues in Entity and create a queue as shown in below figure.

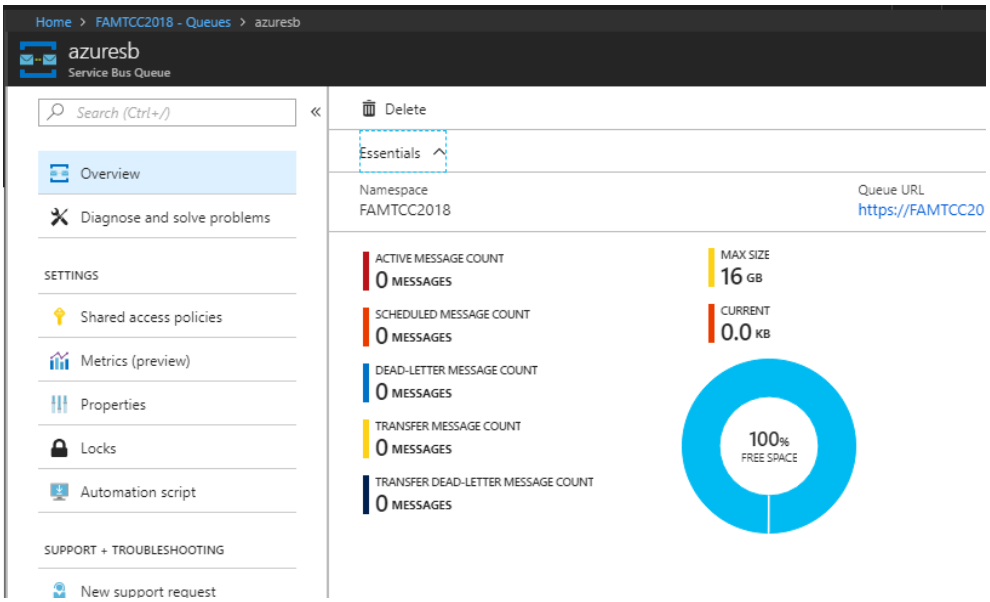

Figure 2

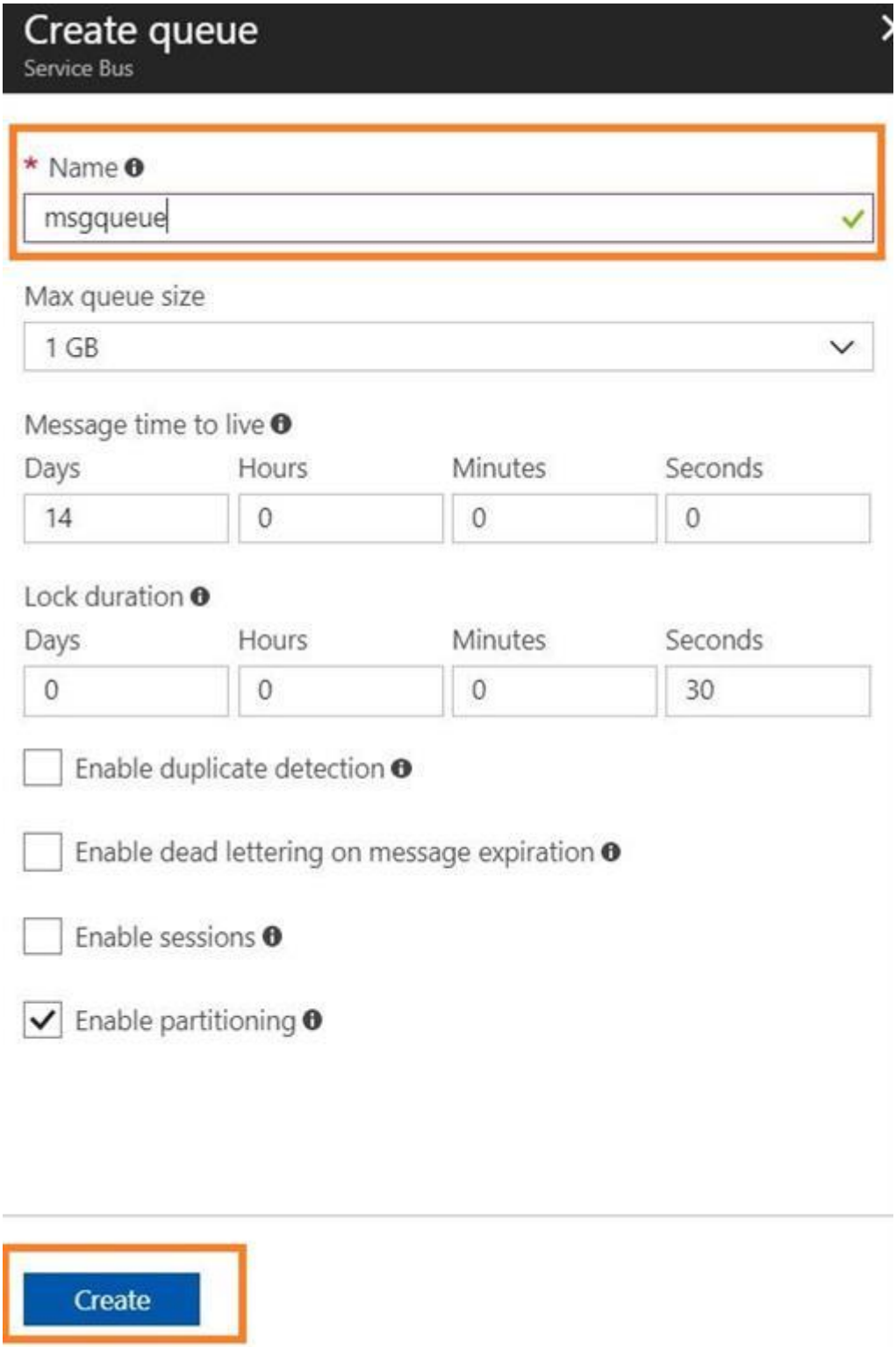

Figure 3

Give a name for the queue, in my case I named it as azuresb.

Message time to live – by default it will be 14 days, it means how long the message will be stored in queue. Keep the rest of the option to default setting and click on create

Send and receive a message using service bus Queues

Create a new console application project in Visual Studio 2017.

Download the service bus queues messaging package from NuGet.

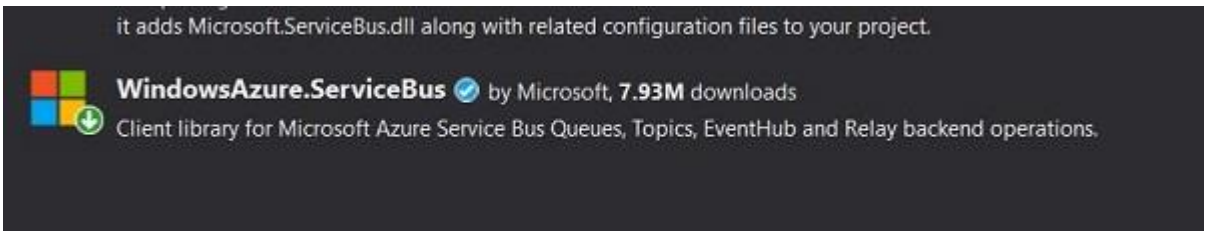

Figure 4: Nuget

Write the below function in Program.cs file.

```
1.  static async Task MainAsync()
2.         {
3.
4.             const string connectionString = "<Namespace connection String>"; // get
    it from azure portal from service bus namespace
    shared                                                                          //
    access policy
5.              const string queueName = "msgqueue";
6.              var _client = QueueClient.CreateFromConnectionString(connectionString, q
    ueueName);
7.              string Message = "I'm in Azure Service Bus Queue";
8.              BrokeredMessage message = new BrokeredMessage(Message);
9.              await _client.SendAsync(message);
10.
11.        }
```

Queue client is an abstract class, where the function CreateFromConnectionString (string, string) is used to create a new copy of Queue Client from the connection string with specified queue path.

BrokeredMessage is a sealed class; here, it is used to initialize the new instance and to serialize the message which is sent to the Queue with the help of queue client. From the above code, it is obvious that the message with string type is serialized and send to queue using Queue Client.

Check the stats in Azure – Shown in Results

Receive a Message

```
1.  static void GetMessage()
2.          {
3.              const string connectionString = "<Namespcae connection string>";
4.              const string queueName = "msgqueue";
5.              var queueClient = QueueClient.CreateFromConnectionString(connectionStri
    ng, queueName);
6.              BrokeredMessage message = queueClient.Receive();
7.              string body = message.GetBody<string>();
8.              message.Complete();
9.              message.Abandon();
10.             Console.WriteLine(body);
11.             Console.ReadLine();
12.
13.         }
```

The above function is used to receive the message from the queue using the QueueClient and get the body of the message using the brokeredMessage.

Program.cs

```
1.  using System;
2.  using System.Threading.Tasks;
3.  using Microsoft.ServiceBus.Messaging;
4.
5.  namespace ServiceFabricDemo
6.  {
7.      class Program
8.      {
9.          static void Main(string[] args)
10.         {
11.             MainAsync().GetAwaiter().GetResult();
12.             GetMessage();
13.         }
14.
15.         static async Task MainAsync()
16.         {
17.
18.             const string connectionString = "Endpoint=sb://msgdemobus.servicebus.wi
    ndows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=rJ0AZa4UFg
    pbUN8fGL7eUJYSLfiwtlvP4mnPAXPSu68=";
19.             const string queueName = "msgqueue";
20.             var _client = QueueClient.CreateFromConnectionString(connectionString,
    queueName);
21.             string Message = "I'm in Azure Service Bus Queue";
22.             BrokeredMessage message = new BrokeredMessage(Message);
23.             await _client.SendAsync(message);
24.
25.         }
26.         static void GetMessage()
27.         {
28.             const string connectionString = "Endpoint=sb://msgdemobus.servicebus.wi
    ndows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=rJ0AZa4UFg
    pbUN8fGL7eUJYSLfiwtlvP4mnPAXPSu68=";
29.             const string queueName = "msgqueue";
30.             var queueClient = QueueClient.CreateFromConnectionString(connectionStri
    ng, queueName);
31.             BrokeredMessage message = queueClient.Receive();
32.             string body = message.GetBody<string>();
33.             message.Complete();
34.             message.Abandon();
35.             Console.WriteLine(body);
36.             Console.ReadLine();
37.
38.         }
39.
40.     }
41. }
```

Output

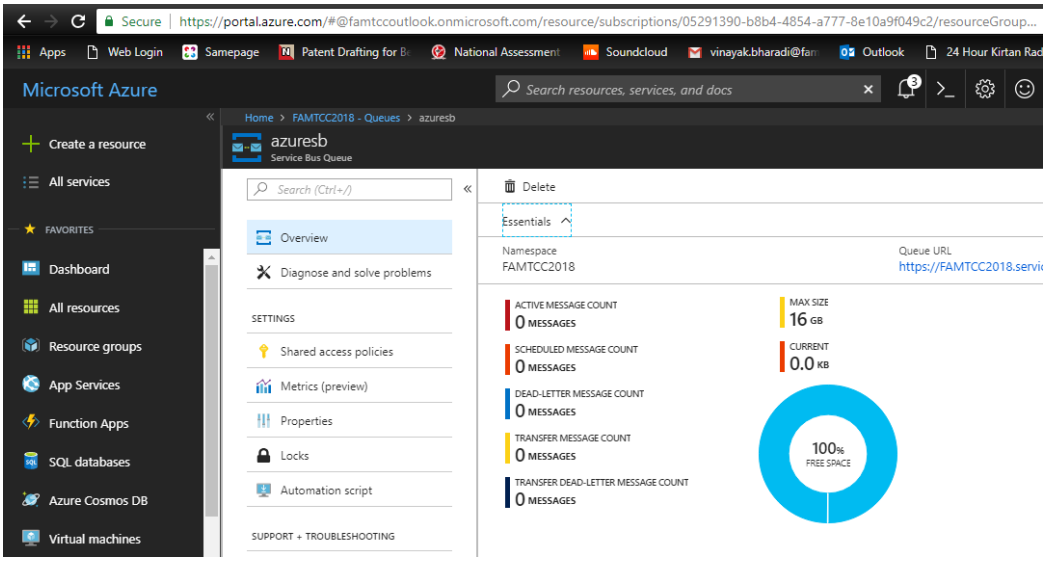

Fig1. Screenshot Showing 0 Messages

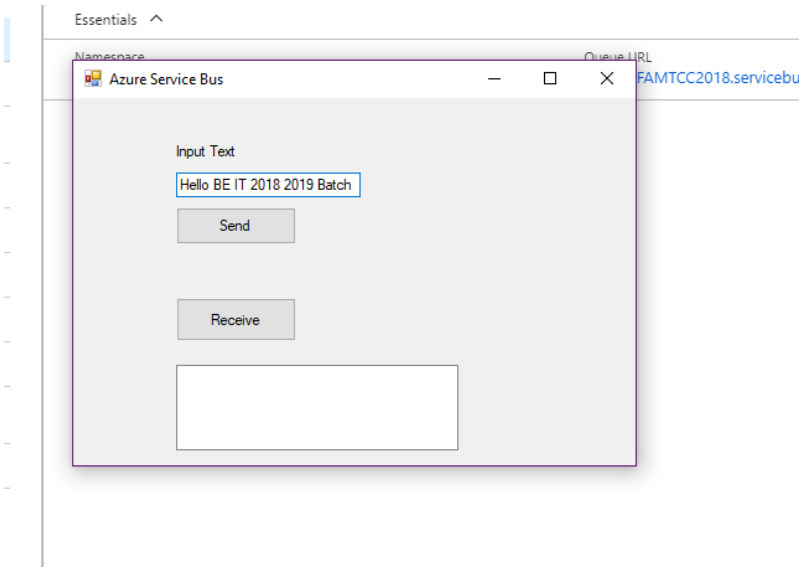

Fig 2. Sending Message on Azure Queue

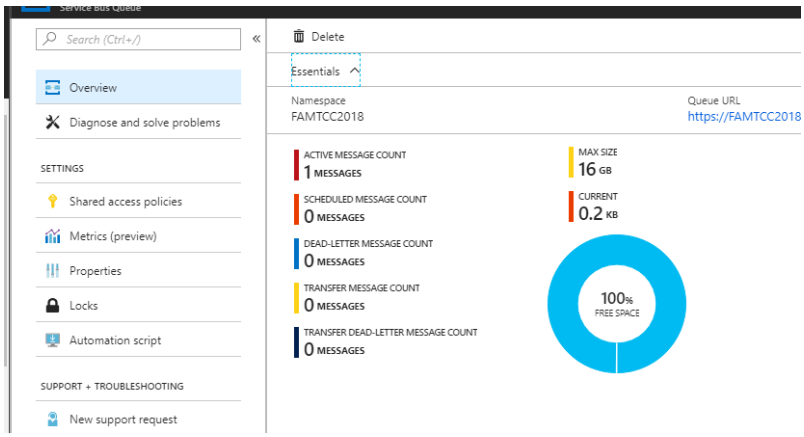

Fig3. Screenshot Showing 1 Messages

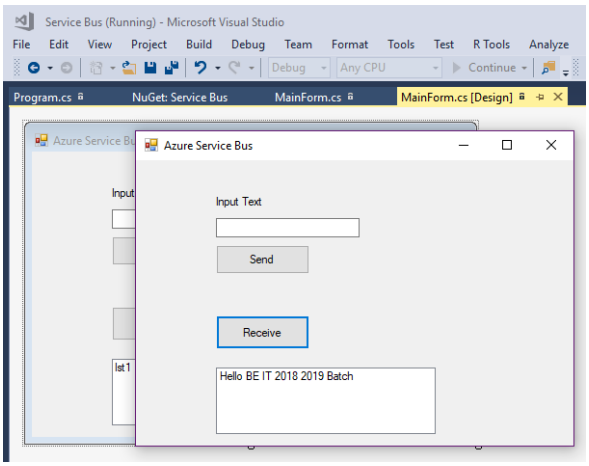

Fig 4. Message received from Queue as shown above

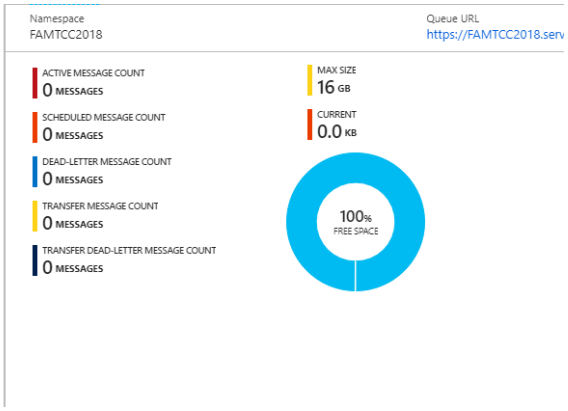

Figure 5 : Message received from Queue – Showing 0 remaining messages

13. Learning Outcomes Achieved

1. We have Configured Azure Cloud Account and Queuing Server
2. We have Configured Azure Message Queue
3. Message was sent using producer program code and consumed by the receiver
4. Discussed about how SaaS Web services transfer messages using queues

14. Conclusion:

1. **Applications of the studied technique in industry**
    a. Message Queues are used by Web services for communication purpose
    b. For invoking web services as well as passing results message queues are used
2. **Engineering Relevance**
    a. SaaS Model is the main consumer of Message Queues
    b. Cloud based applications are using message ques for sending and receiving messages
3. **Skills Developed**
    a. Configuration of message queues and server
    b. Generating and Consuming messages on message queues

References :

[1] https://www.rabbitmq.com/

[2] https://en.wikipedia.org/wiki/Message_queuing_service

Viva Questions

4. What is SaaS Model?
5. List components of Message Queuing Server?
6. Which version of visual studio is used for this experiment?

Teachers Interaction with Students

1. Concept of Message Queuing
2. Rabbit MQ/ Azure Service Bus Configuration
3. Coding for Generating and Consuming Messages