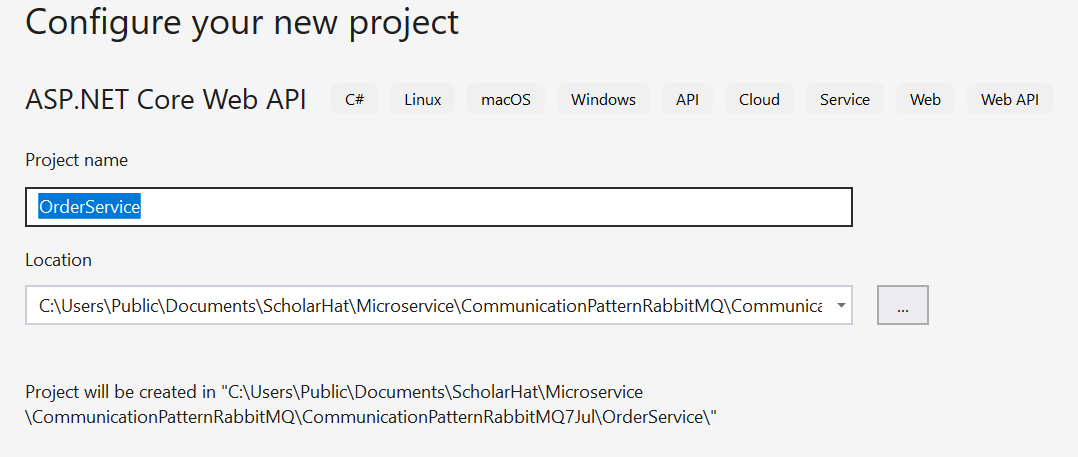
[**1. SECTION – 1 : Communication Pattern - RabbitMQ** 1](#_Toc171759164)

# **1. SECTION – 1 : Communication Pattern - RabbitMQ**

**Folder Structure Creation:**

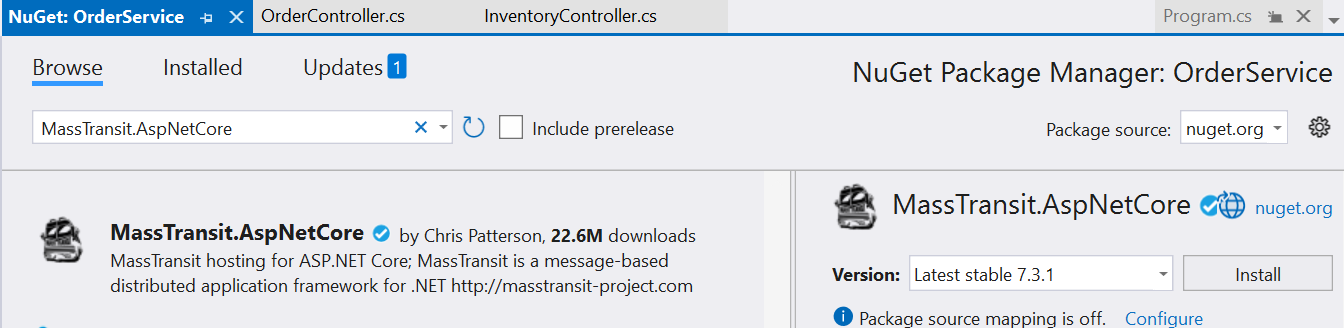
1. Create New Project 🡪 select blank solution template 🡪 Give a name to empty solution as CommunicationPatternRabbitMQ7Jul and select folder path.
2. Under Empty solution add Asp.Net Core Web API Template as new Project for InventoryService and OrderService respectively.

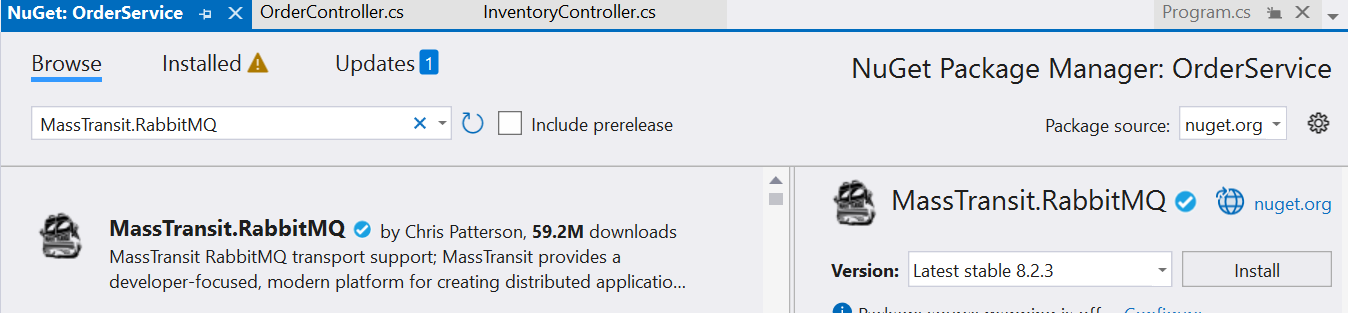


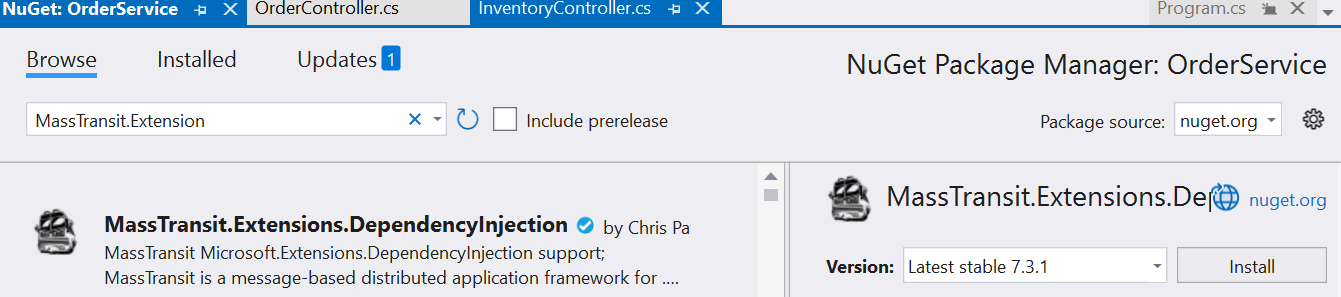
A screenshot of a computer

Description automatically generated

1. Add InventoryController in InventoryService and OrderController in OrderServisce respectively.
2. Add MassTransit.AspNetCore , MassTransit.RabbitMQ and MassTransit.Extensions.DependancyInjection as a NugetPackage in both OrderService and InventoryService respectively.







1. Add the ServiceBus Setting in appsetting.json file of OrderService and add the Services to the container in Program.cs file of OrderService.

"ServiceBus": {

"Uri": "rabbitmq://localhost",

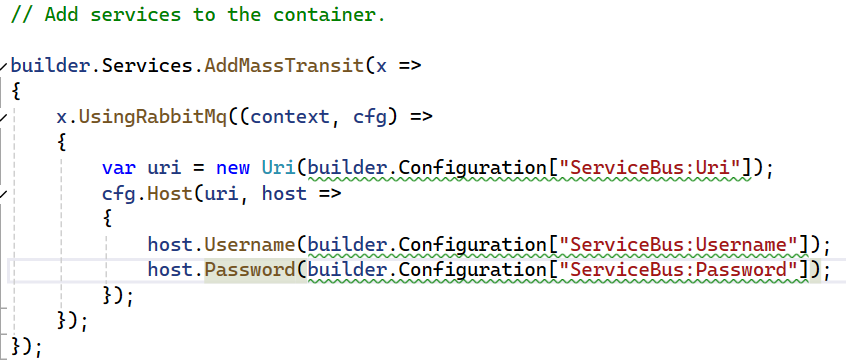
"Username": "guest",

"Password": "guest",

"Exchange": "order\_exchange",

"RoutingKey": "order\_routing\_key"

}

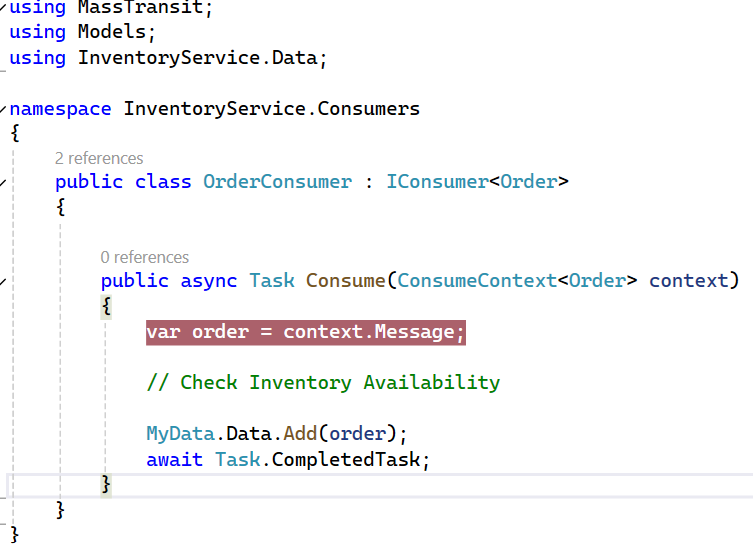


1. Add the ServiceBus Setting in appsetting.json file of InventoryService and service to the container in Program.cs file of InventoryService.

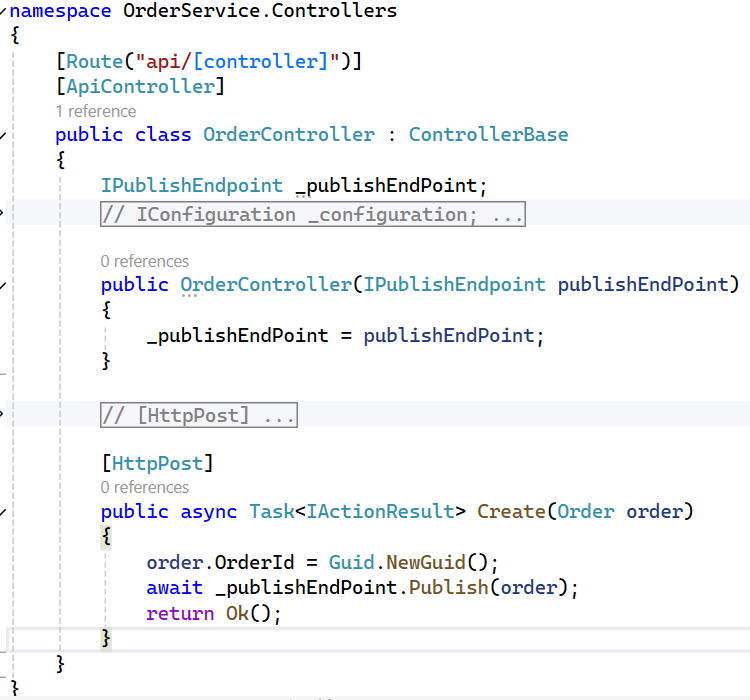




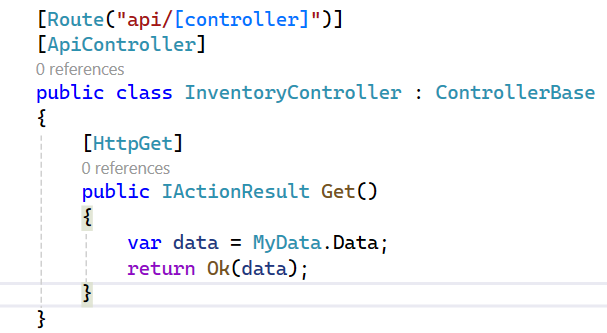
1. Add OrderConsumer class inside Consumers folder of InventoryService which consumes the messages.



1. OrderController of OrderService



1. InventoryController of InventoryService.

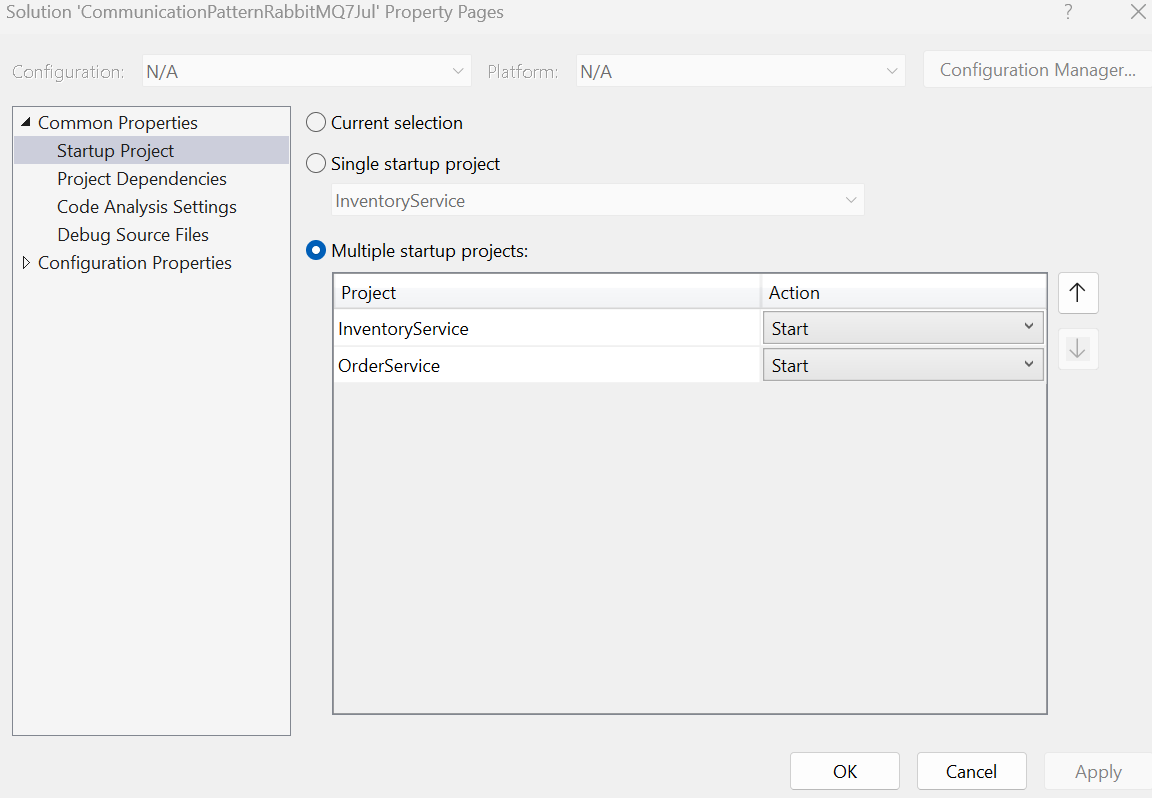


Create the data property for in memory collection to hold the data that are sent from order service and collect the message from InvetoryConsumer for test the messages in RabbitMQ.

A computer screen shot of a computer code

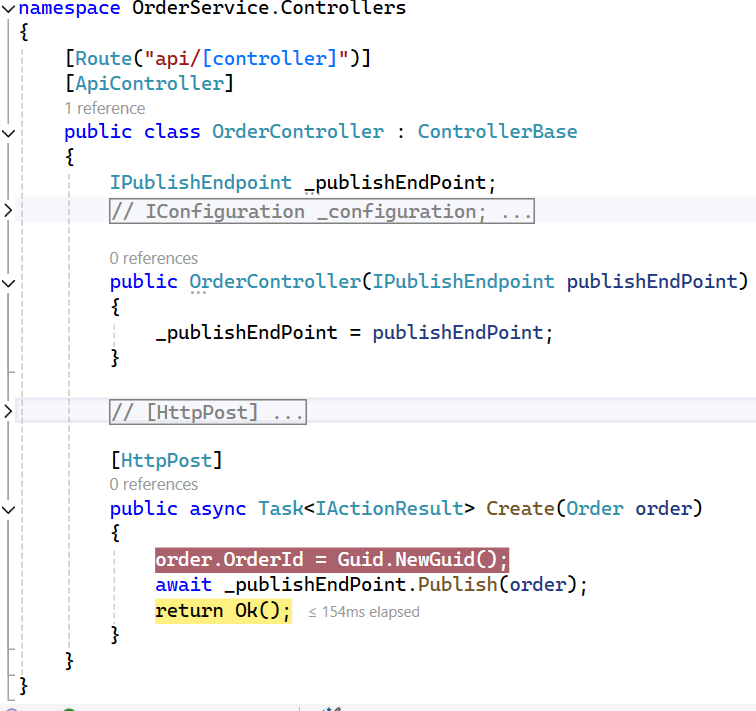
Description automatically generated

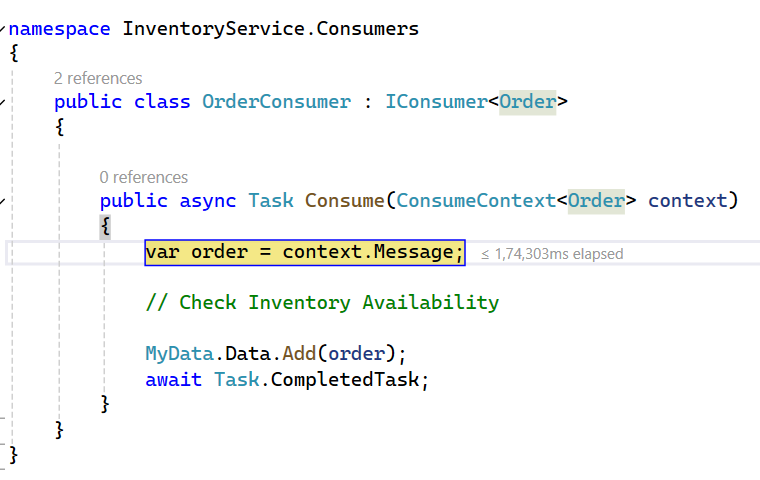
1. Setup both the project as statup project and run.
2. Run the RabbitMQ (Either docker / Setup )
3. Post the order details from the Swagger, add debug in InventoryController action method of InventoryService and check the message in RabbitMQ. It shows the graph in RabbitMQ and then consume the message through InventoryController action method.

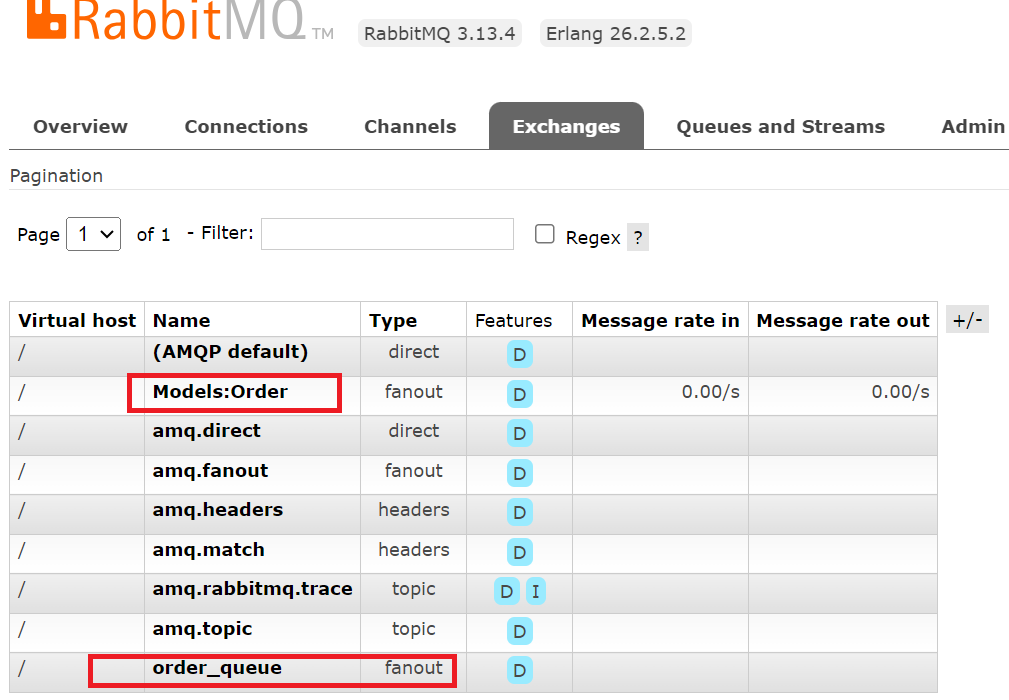


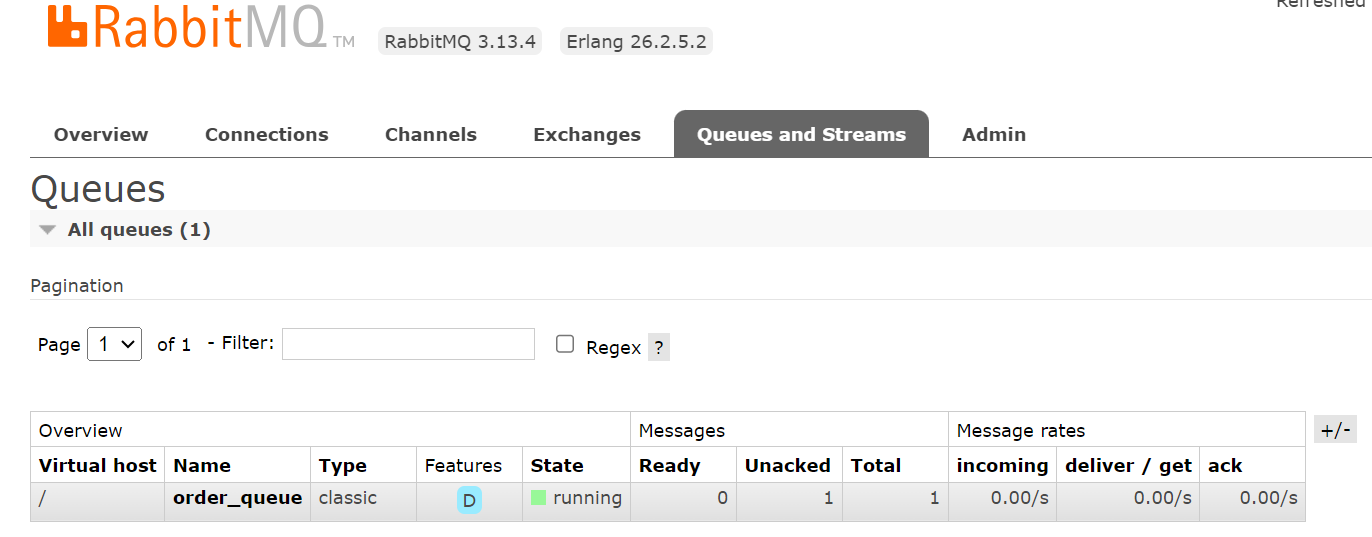
A screenshot of a phone

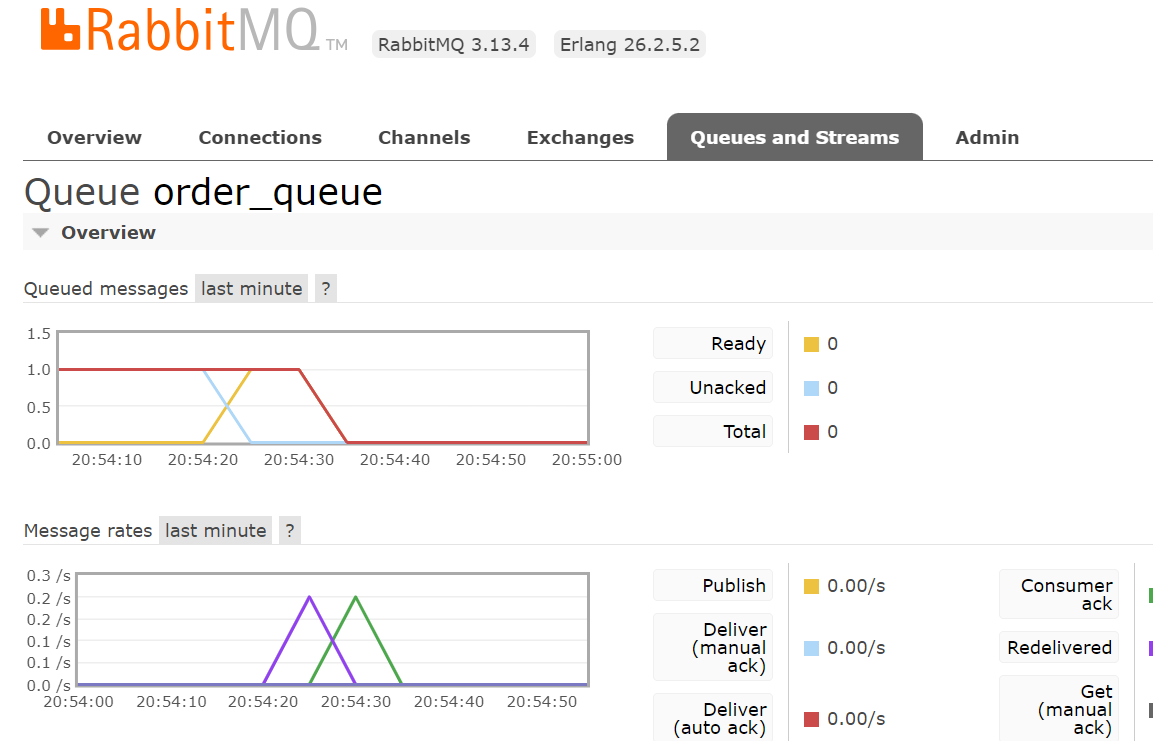
Description automatically generated





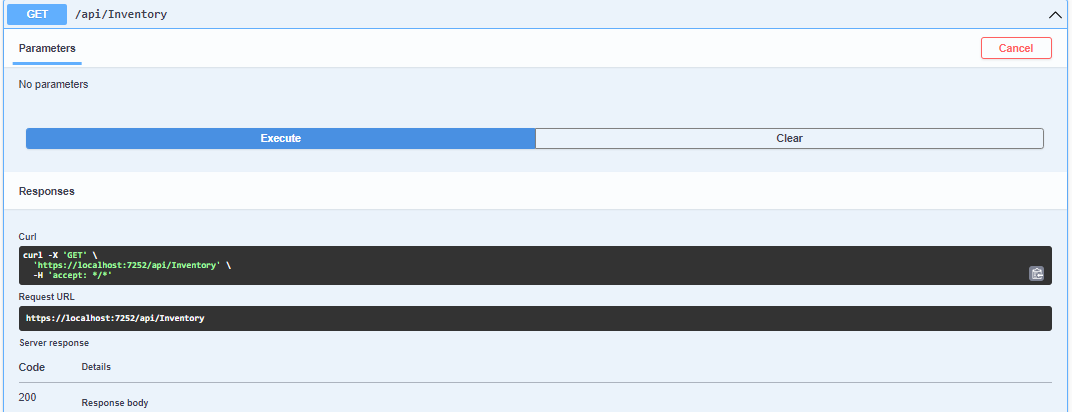






A screenshot of a computer

Description automatically generated

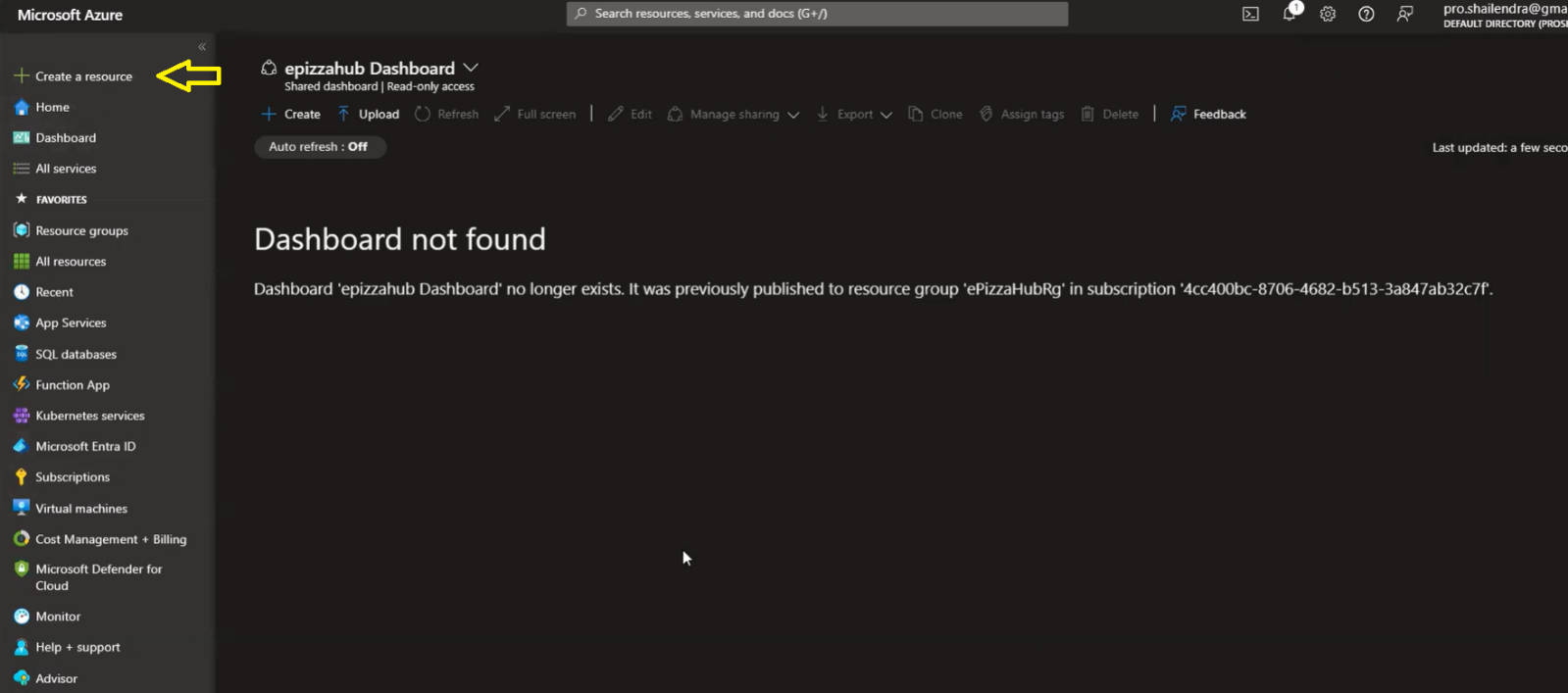


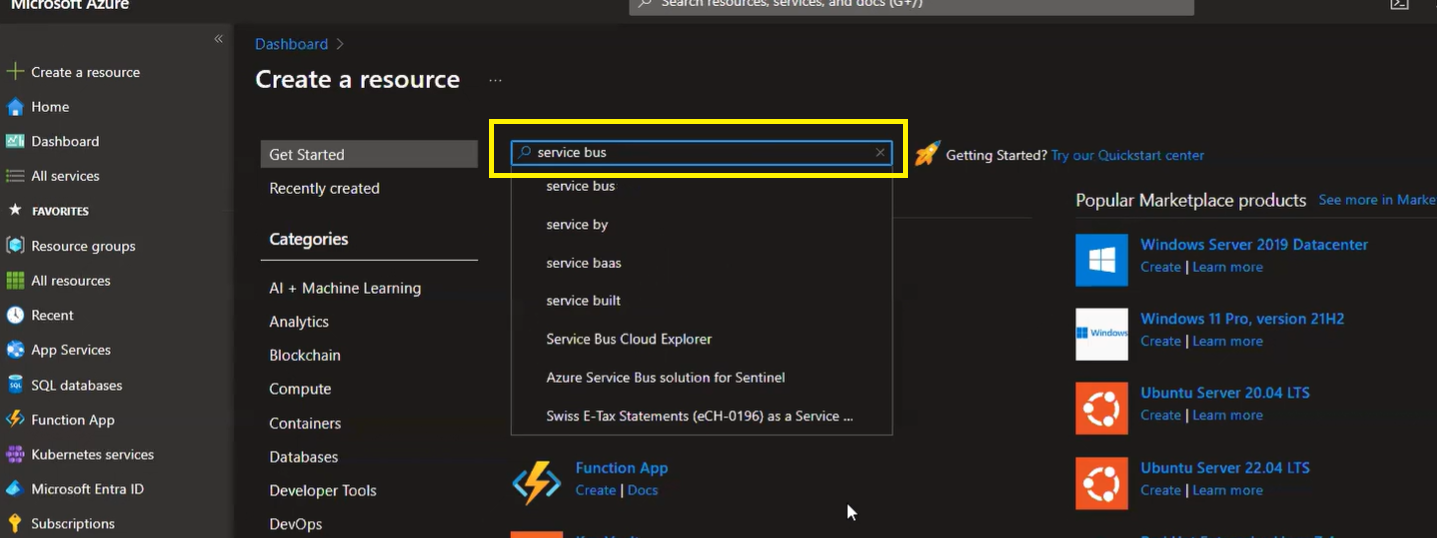
A black and white stripes

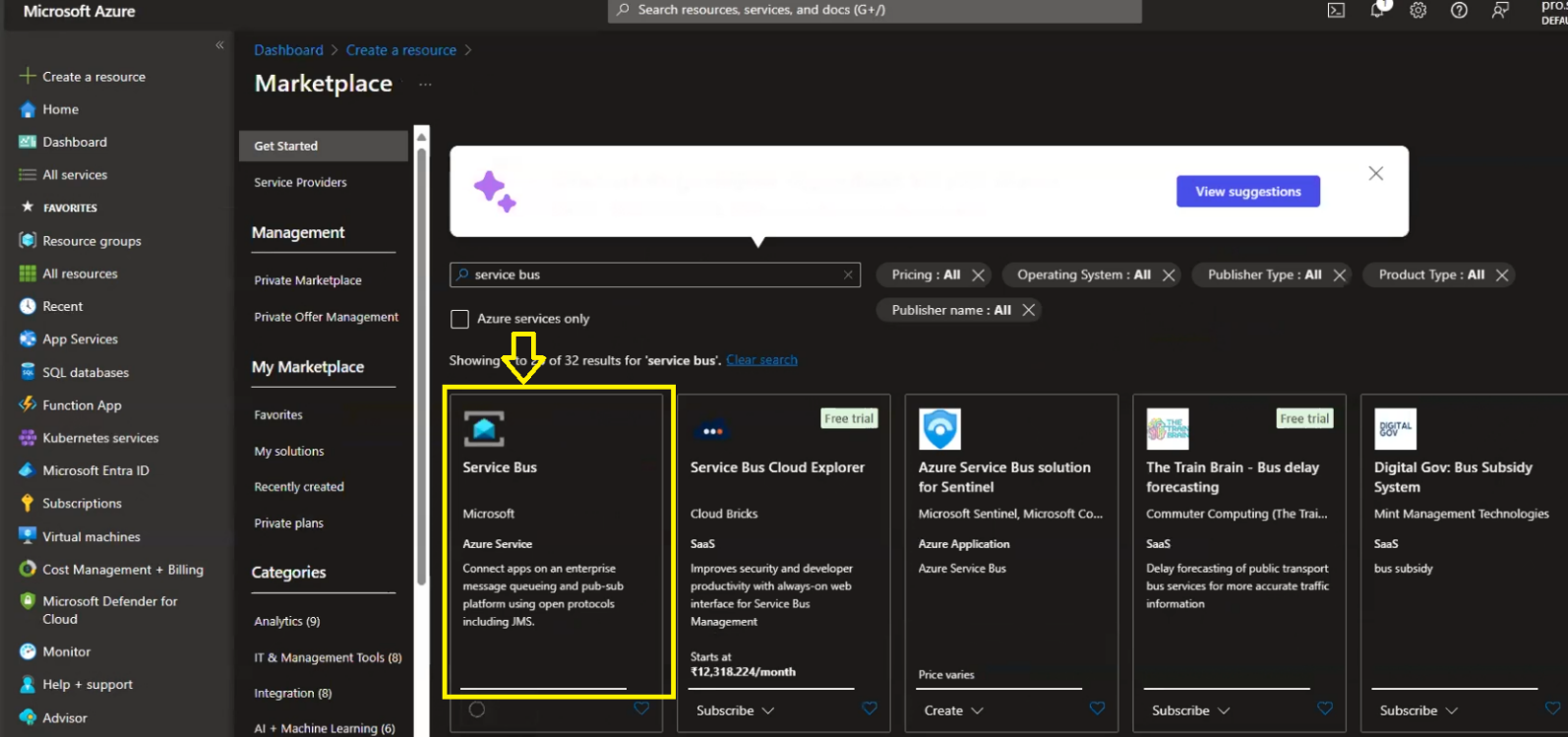
Description automatically generated

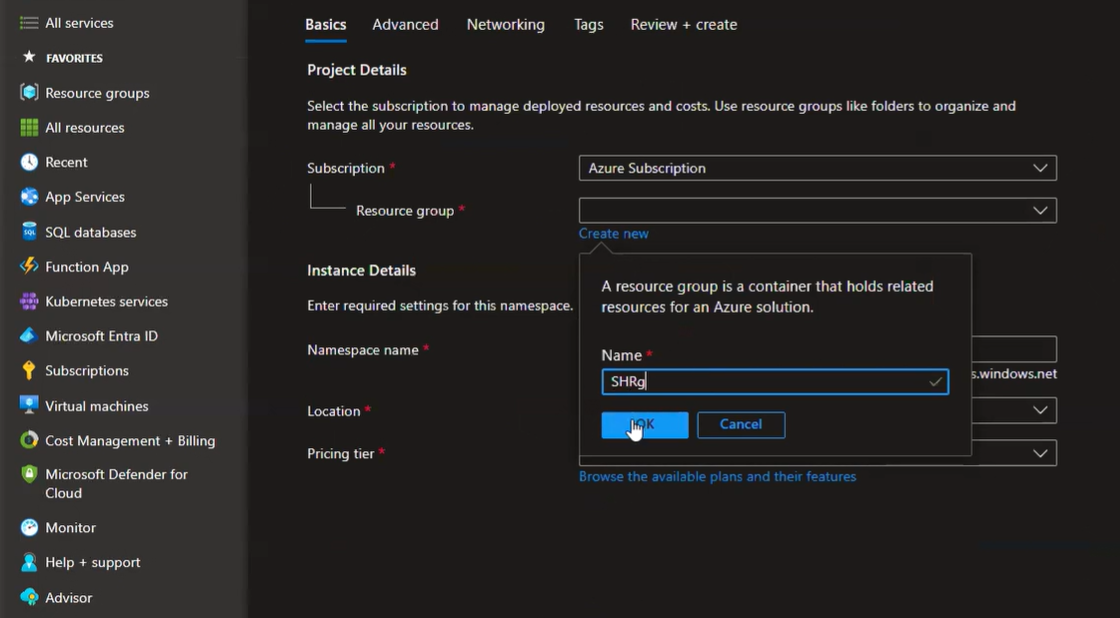
# **2. SECTION – 2 : Communication Pattern – Azure Service Bus**

1. Login to <https://portal.azure.com>



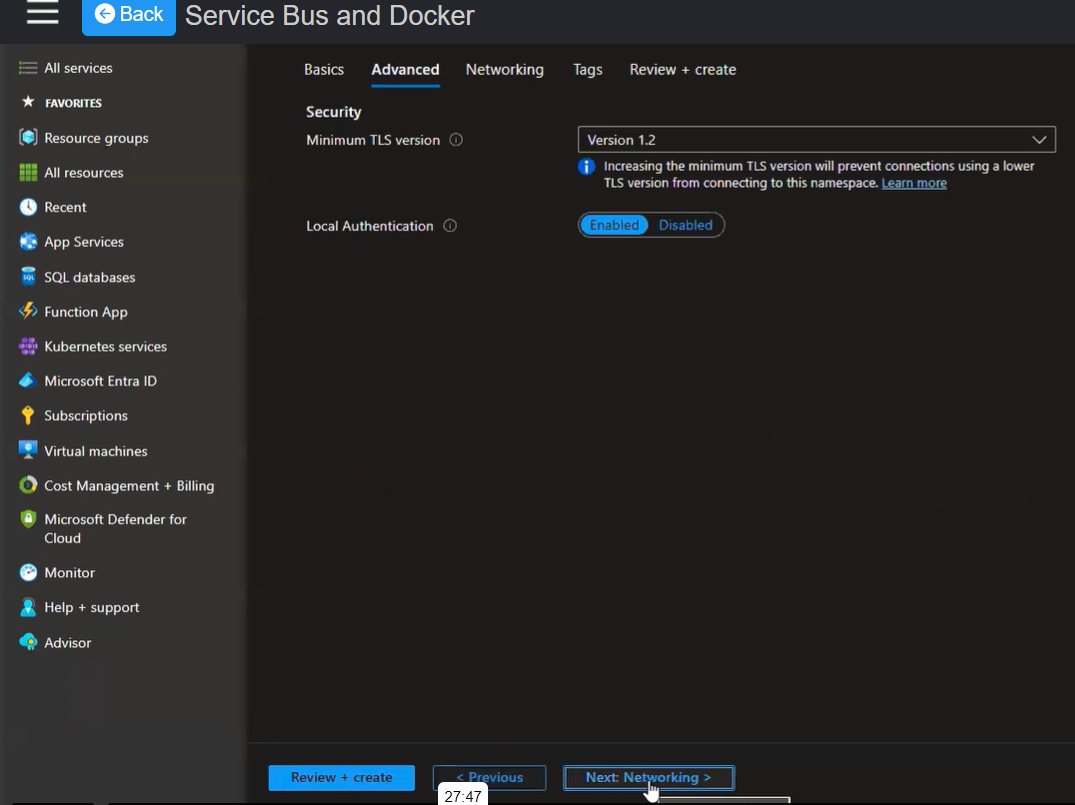






A screenshot of a computer

Description automatically generated

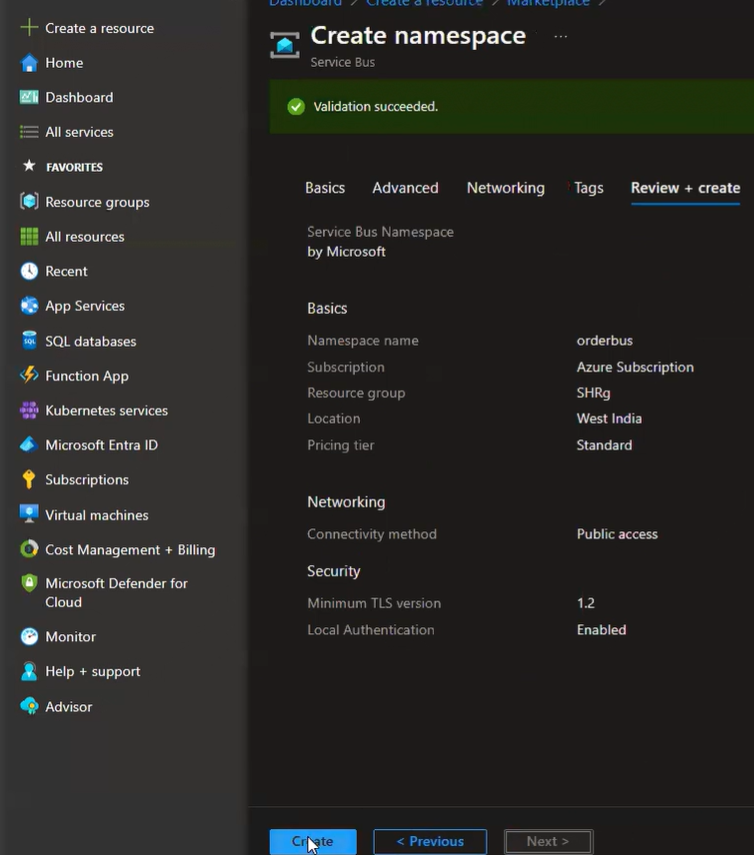


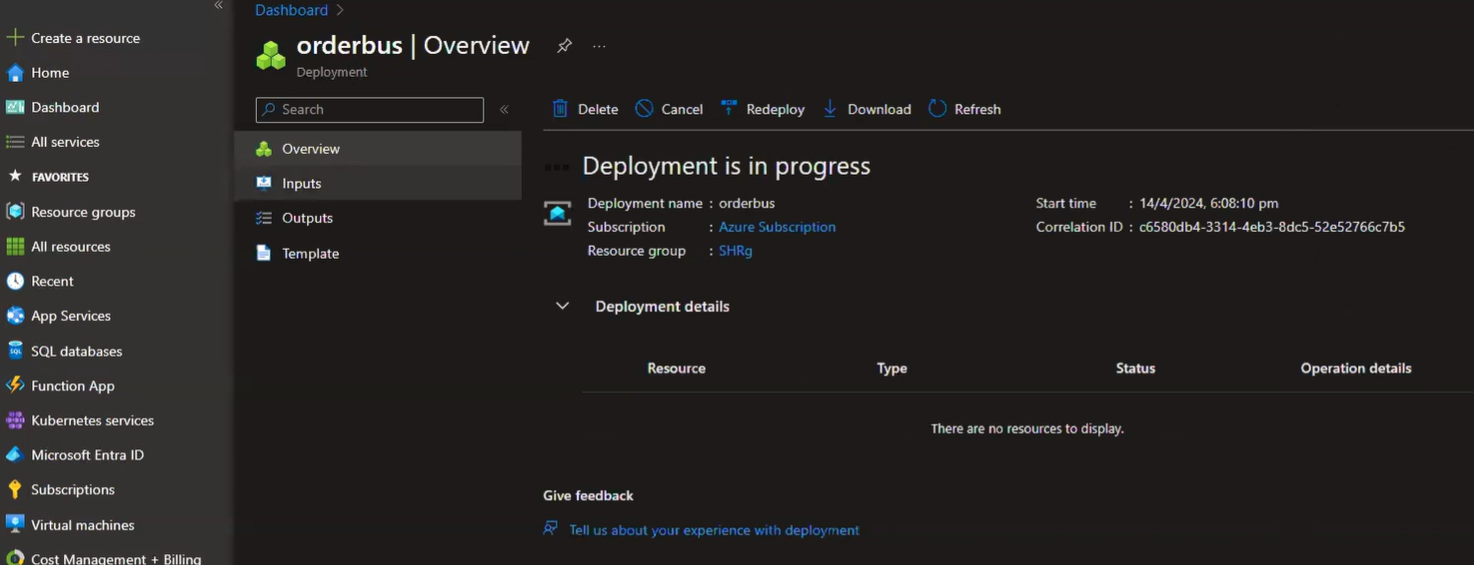
A screenshot of a computer

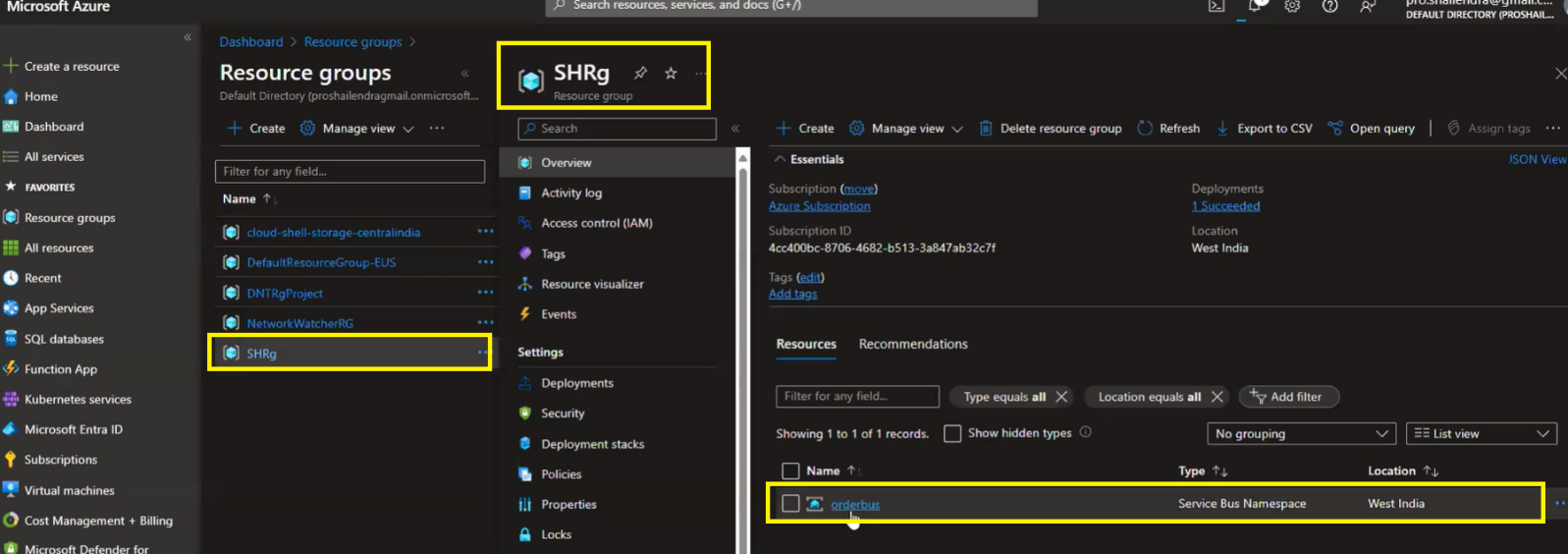
Description automatically generated

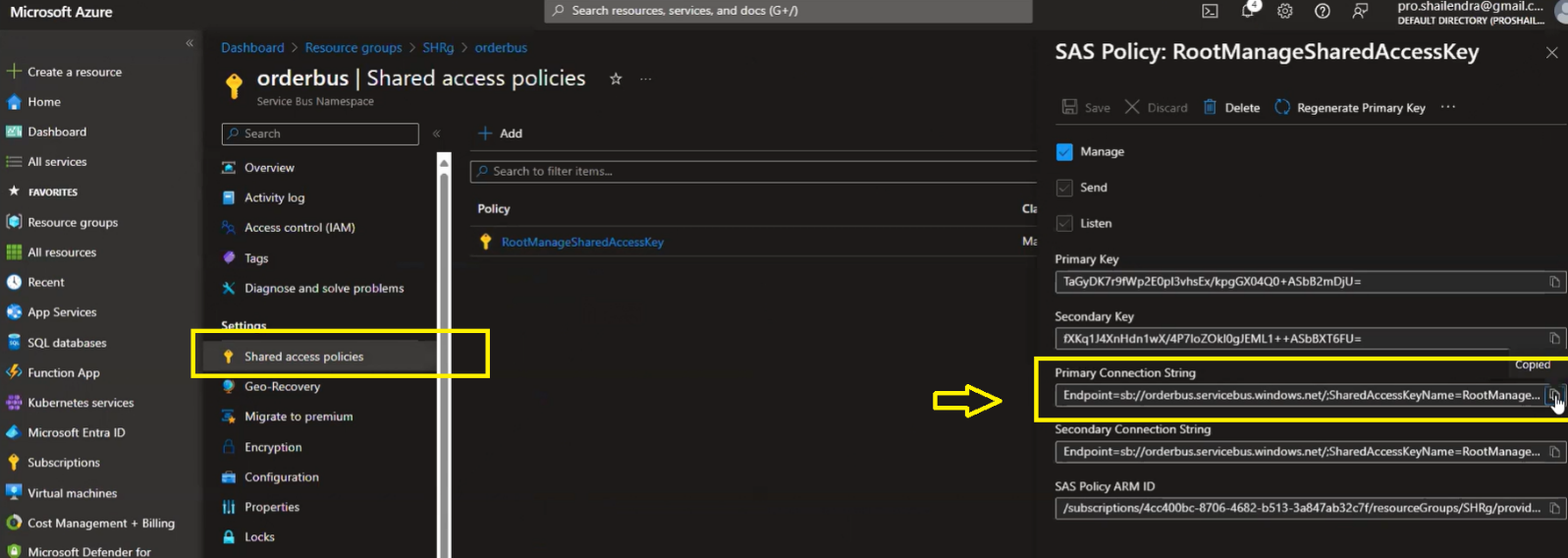
A screenshot of a computer

Description automatically generated









1. Create New Blank Solution for Azure Service Bus named as ServiceBusCommunication and under Empty solution add 2 New Projects (ASP.Net Core Web API Template) named it as OrderSerivce and InventoryService respectively same as we did in the RabbitMQ communication pattern.
2. Define the ServiceBus configuration in appsetting.json file of InventoryService and OrderService.

"ServiceBus": {

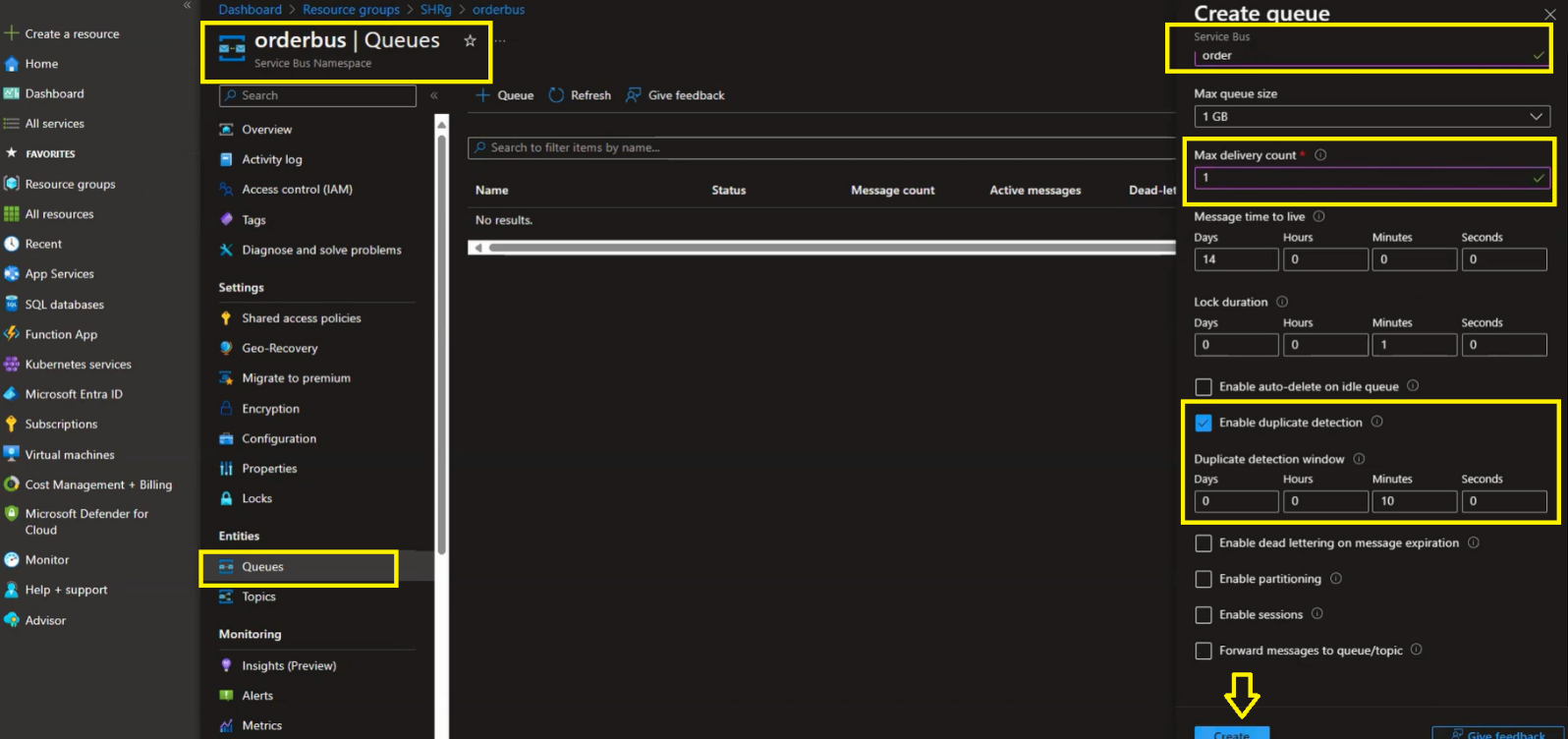
//Azure Service Bus Connection String

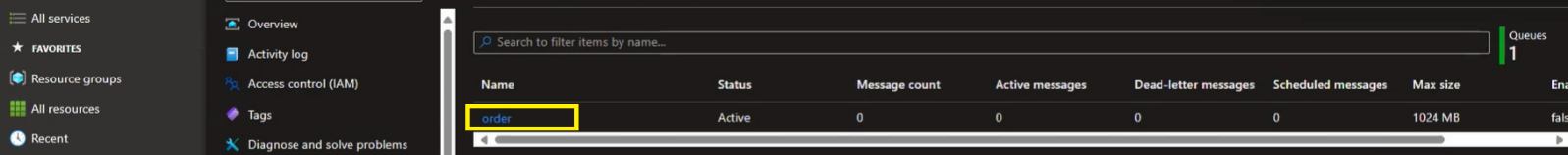
"ConnectionString": "Endpoint=sb://orderbus.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=TaGyDK7r9fWp2E0pl3vhsEx/kpgGX04Q0+ASbB2mDjU=",

"QueueName": "order"

}

1. Create a Queue in Azure portal for orderbus (Service Bus Namespace)





A screenshot of a computer

Description automatically generated

1. Add a below Nuget Package for Azure Service Bus in both InventoryService and OrderService. We can also use MassTransit, but Azure.Messaging.Service bus full fill the request hence ne need to use MassTransit in Azure Service Bus communication pattern.

A screenshot of a computer

Description automatically generated

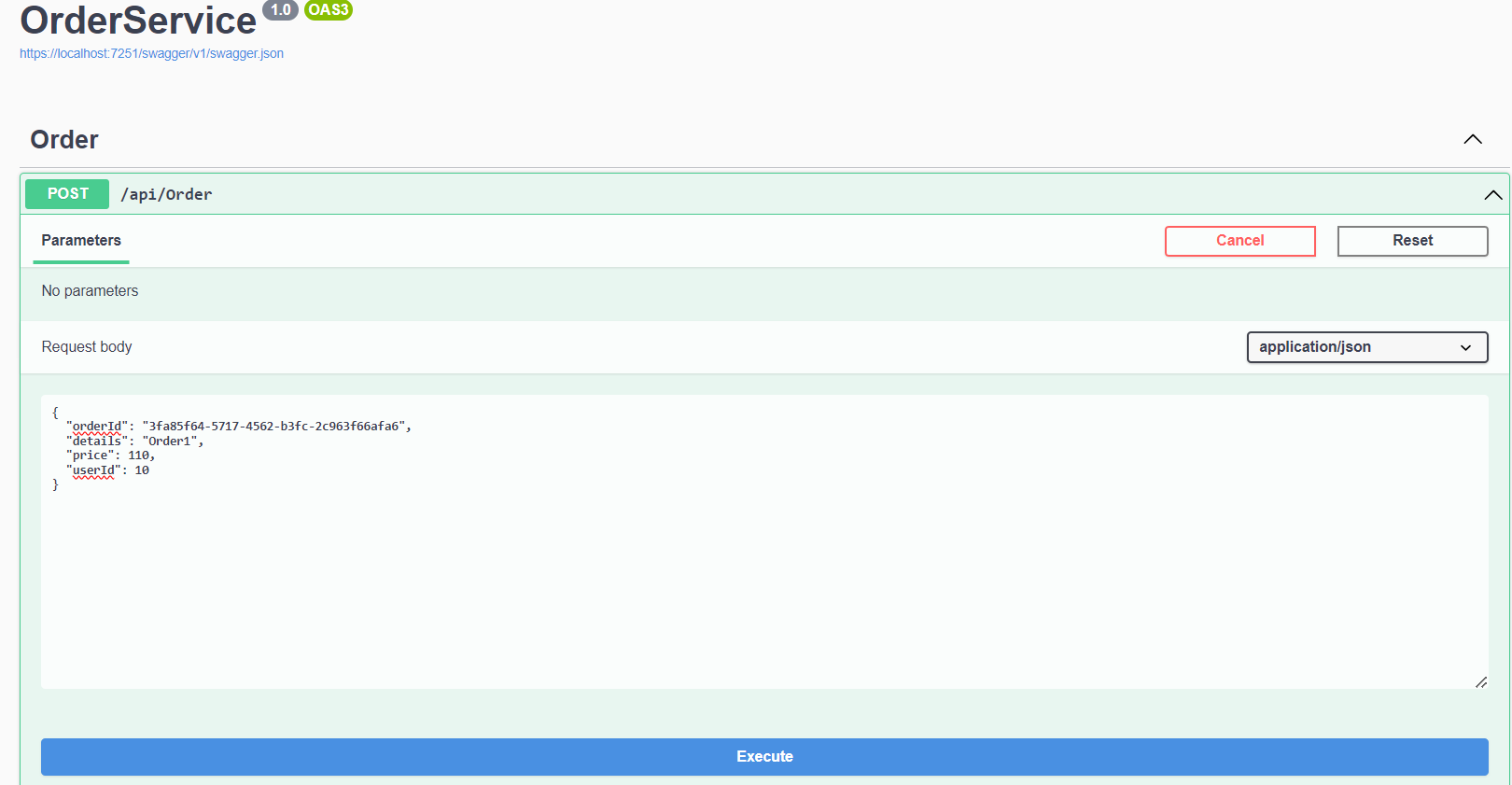
1. Add Api Controller and Models in OrderService.
2. Add Controller,Consumers,Model and Data folder in InventoryService.
3. As soon as message send, the consumer will receive the notification, for this purpose we need to add Service in Program.cs file of InventoryService.

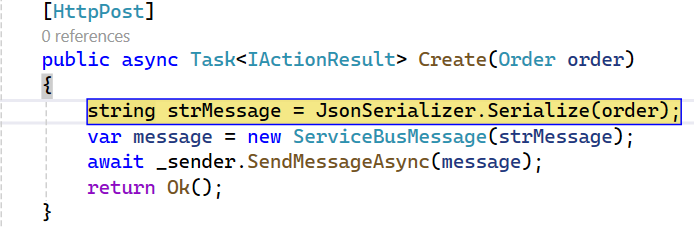
builder.Services.AddSingleton<IOrderConsumer, OrderConsumer>();

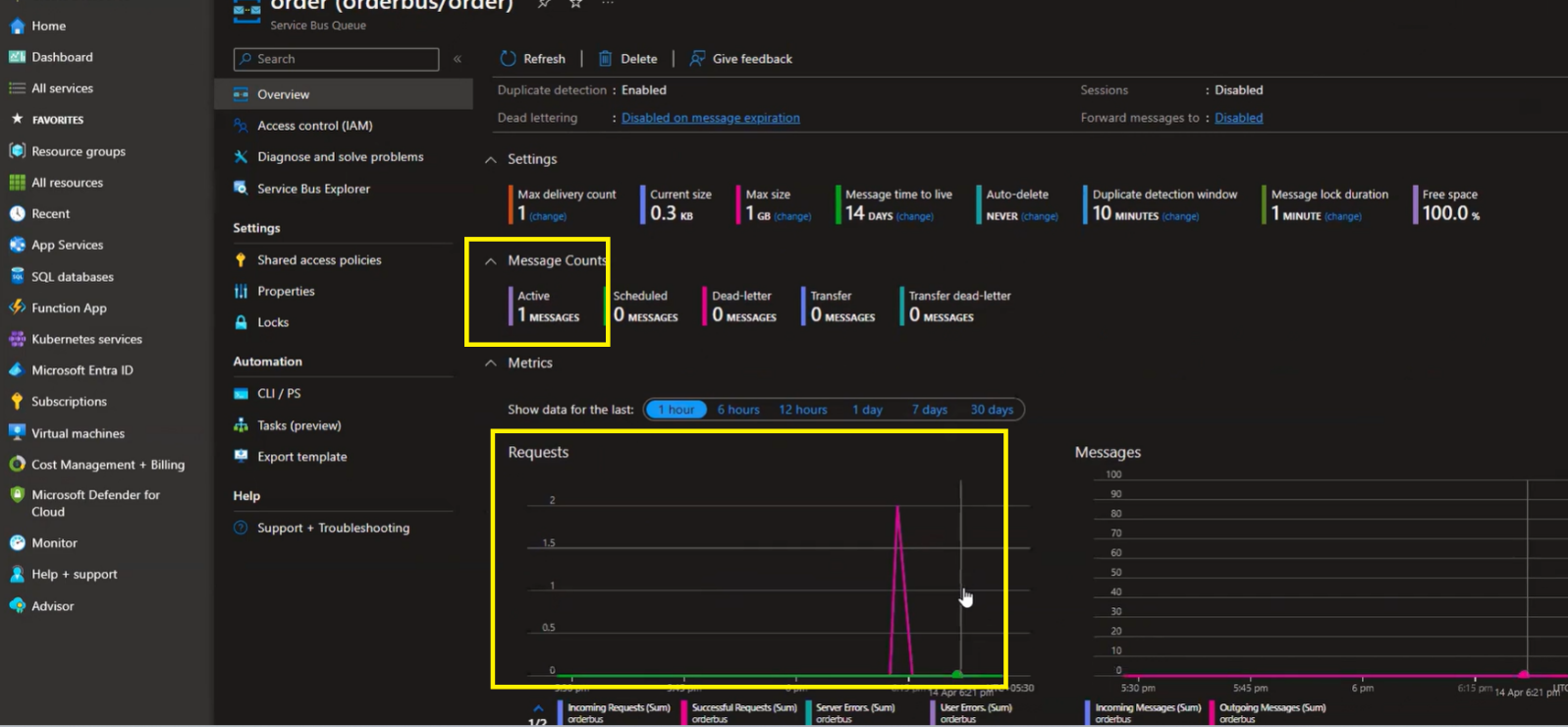
var bus = app.Services.GetService<IOrderConsumer>();

bus?.RegisterReceiveMessageHandler();

1. Setup both the Project as statup projects and Run. Post the Order in swagger api.







1. In OrderConsumer hit the debug and Inventory Swagger api, get the response as below.

