**How to create & publish Azure functions in Java.**

In this article, I am going to explain how to create, publish Azure functions and integrate with SQL data base in java.

## You Will Learn

How can we create and publish azure functions by using java

1.HttpTrigger Function

2.QueueTrigger

3.BlobTrigger

4.TimeTrigger

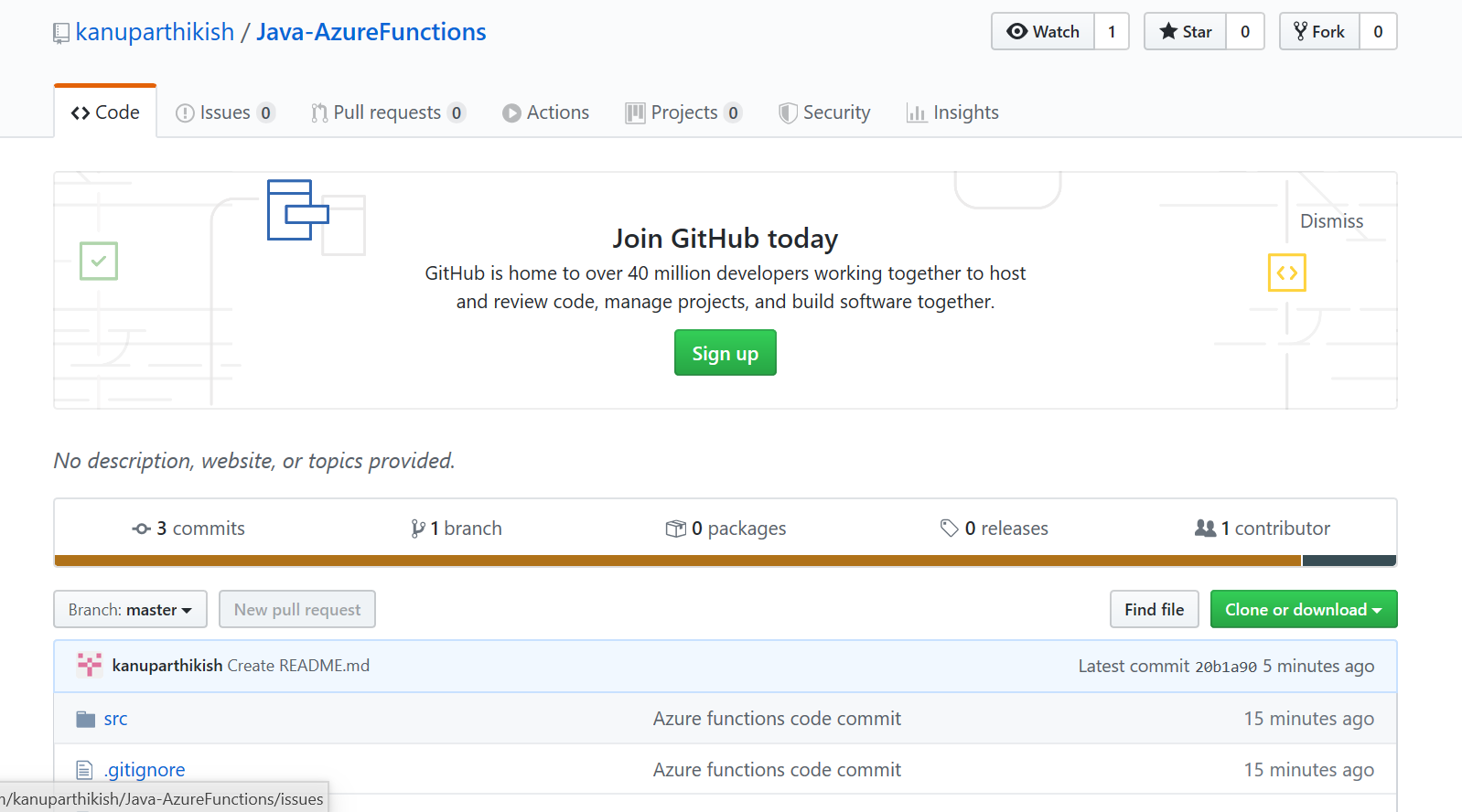
5. HttpTrigger Function integrate with Azure SQL

## Tools Required

* Maven 3.0+ build tool
* Eclipse as IDE.
* Postman
* Azure Subscription

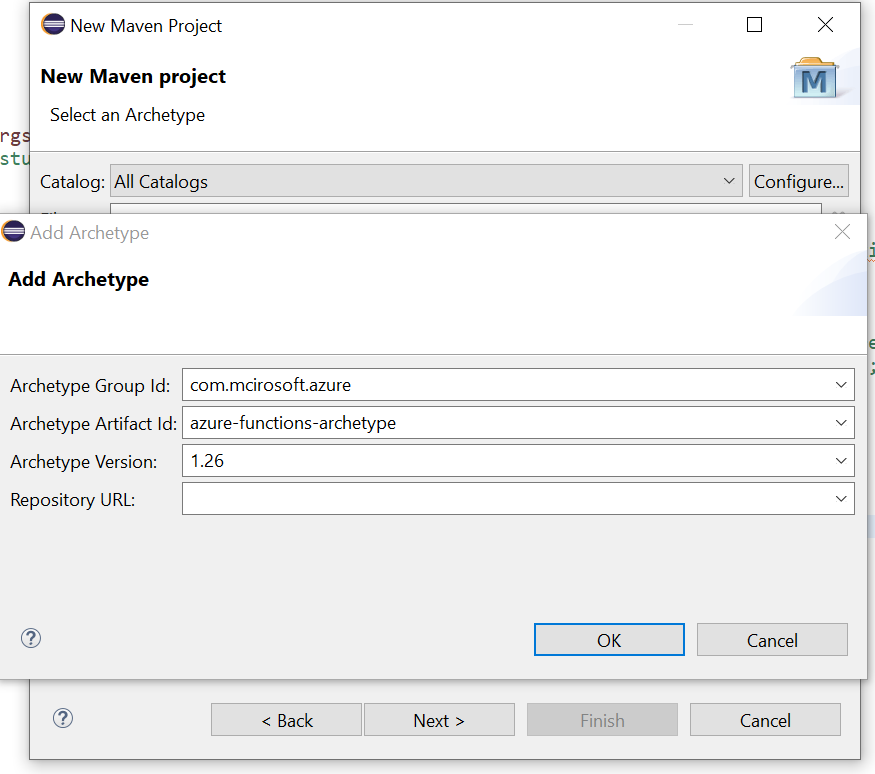
## Complete Maven Project with Code Example is available in Github

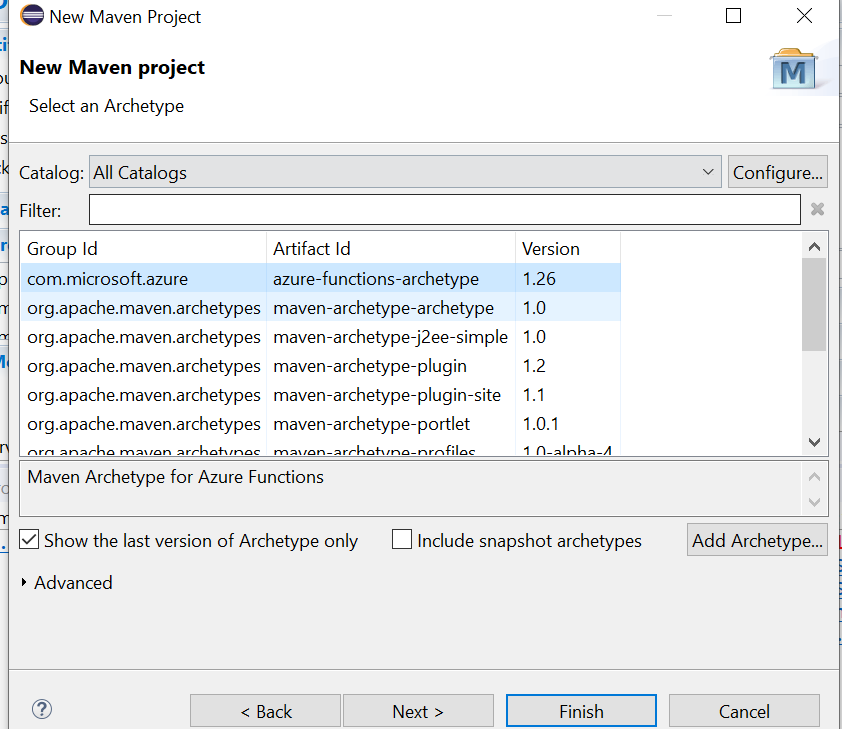
<https://github.com/kanuparthikish/Java-AzureFunctions.git>

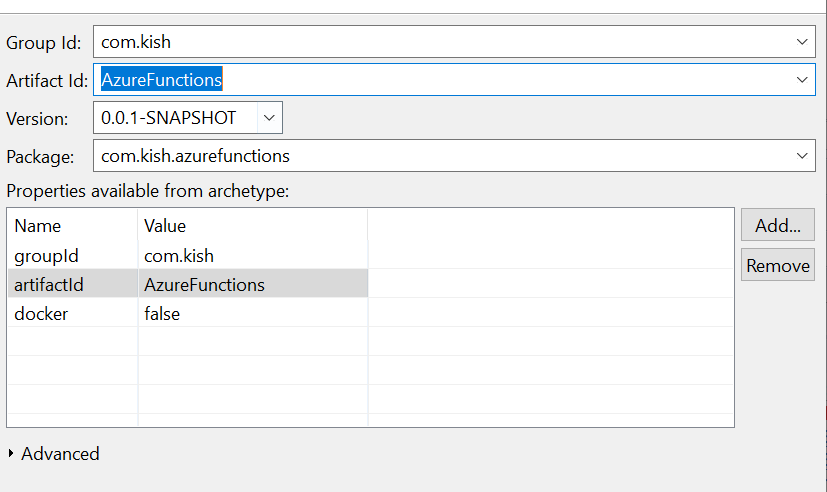


1. Add new Arch type while creating new maven project. Please refer the following link for latest version

<https://mvnrepository.com/artifact/com.microsoft.azure/azure-functions-archetype>





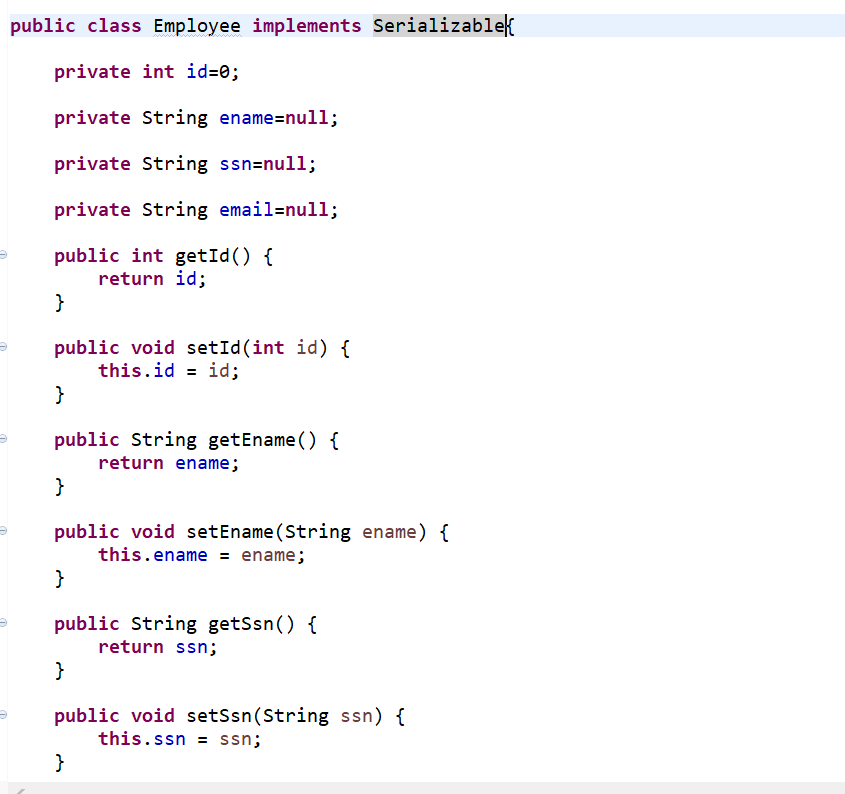


**Http Trigger**

2. Create the Http Trigger Function which reads request parameter and request body as simple Employee POJO and sends the response as HTTP status OK with request parameter value and employee object name.

In case if request parameter value is null then sends response as Http Status Bad request.

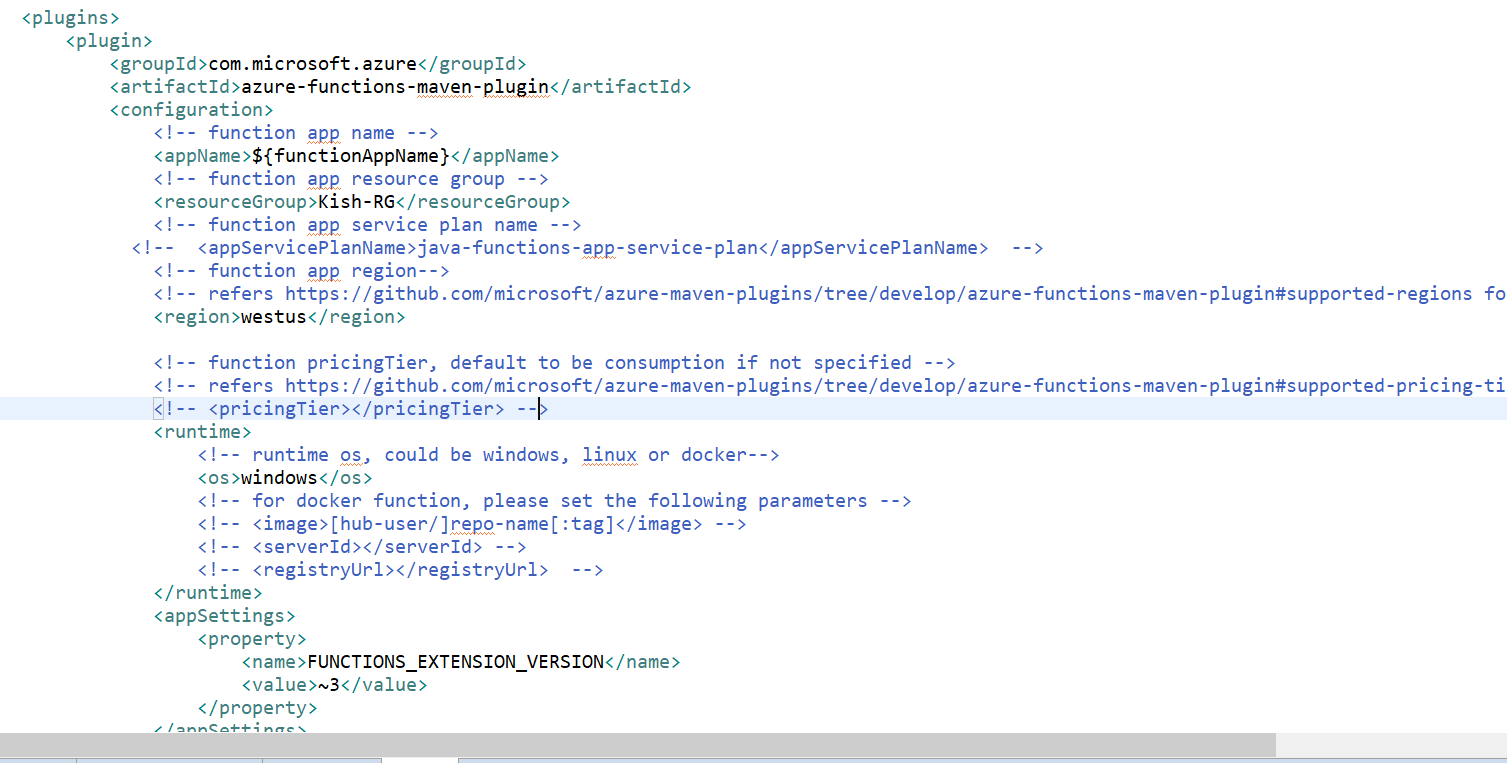




3. in POM.xml update function App name as “kishazureappfunction”

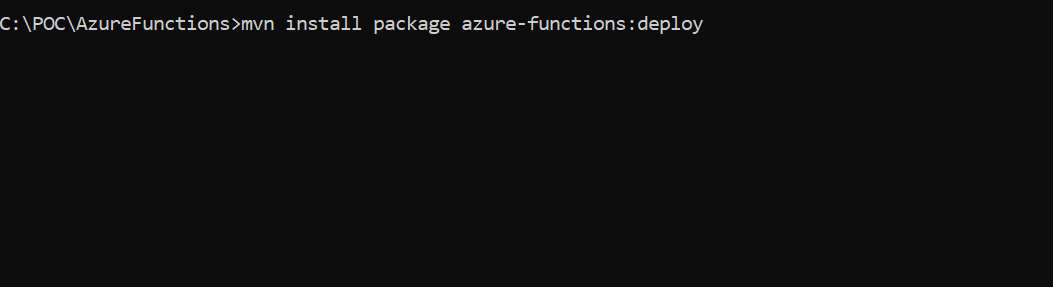


4. update resource group name ,region, os and App service plan

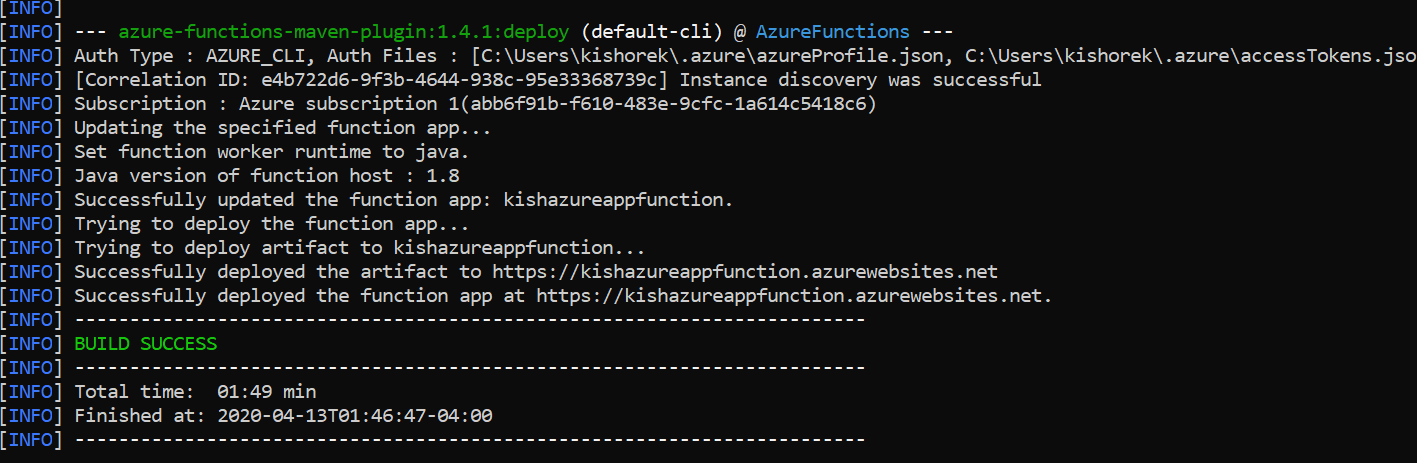


5. To Deploy the Azure function use the following command

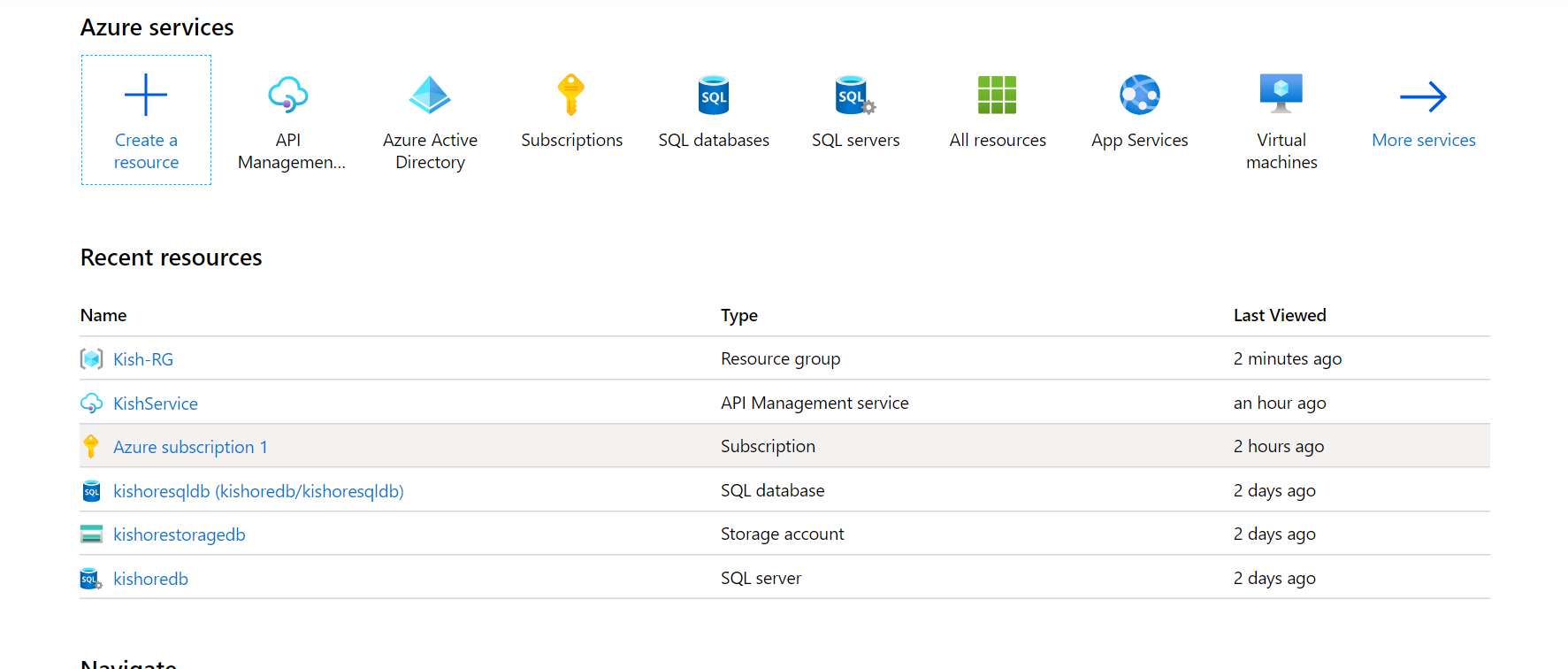
mvn install package azure-functions:deploy

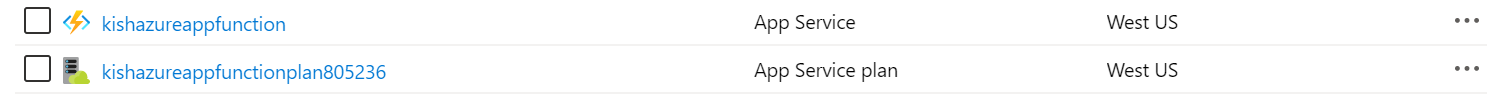


The function is published sucessfully.



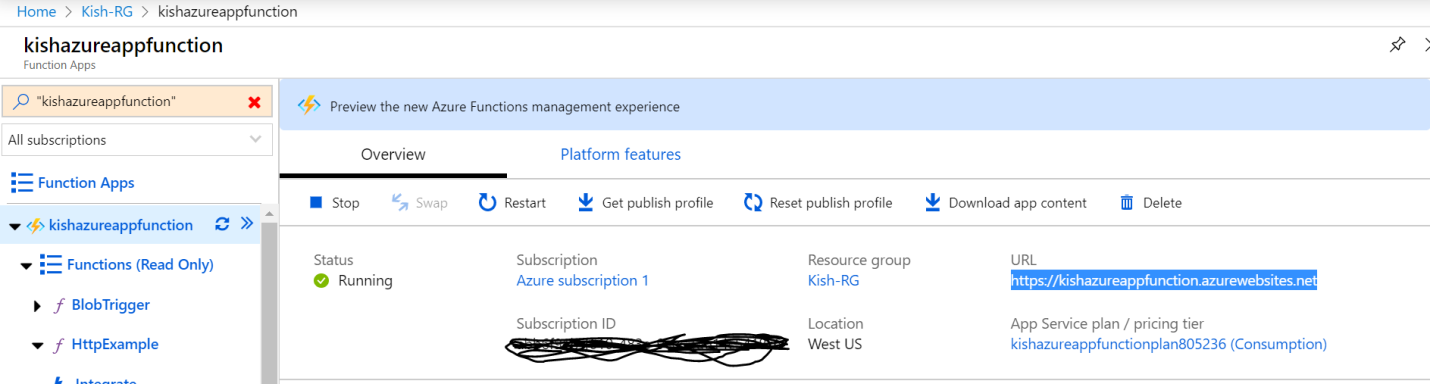
6. Log on to the Azure Web portal and under resource group you can find the azure app function and asp.



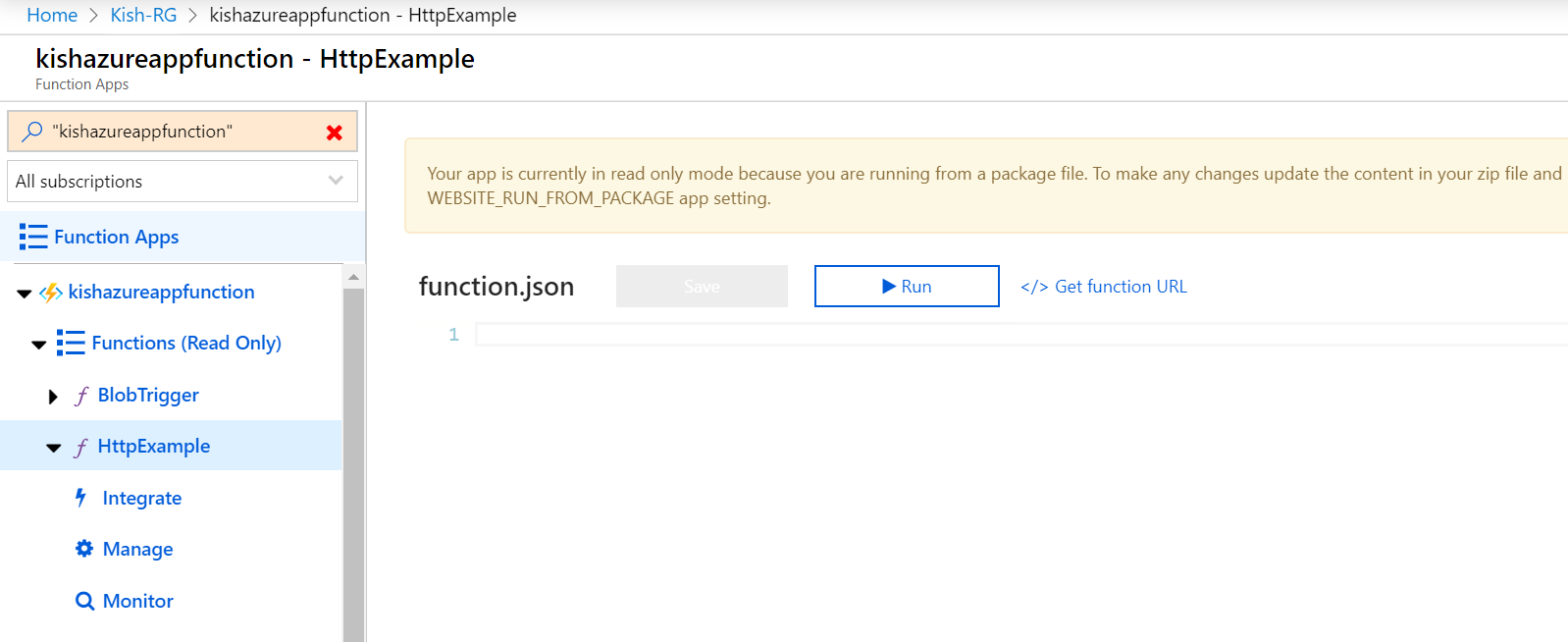


7. Navigate to the appfunction

The app function published and url is <https://kishazureappfunction.azurewebsites.net>

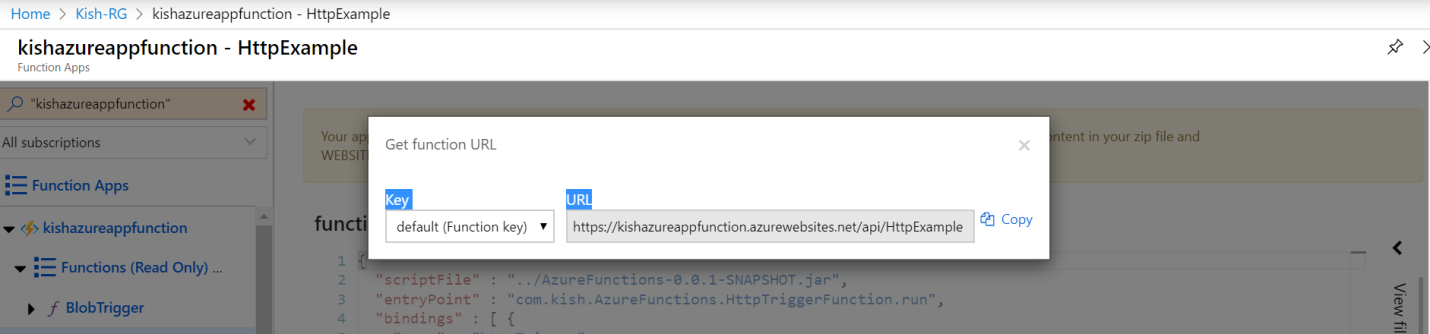


Navigate to the Http Example



The HttpExample function listens on api/HttpExample and the URL is

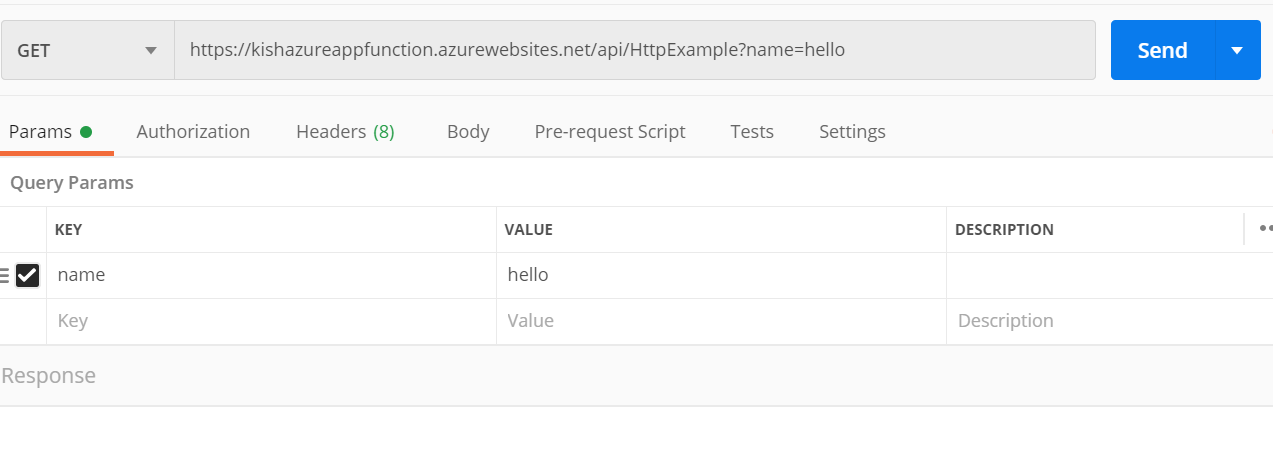
<https://kishazureappfunction.azurewebsites.net/api/HttpExample>



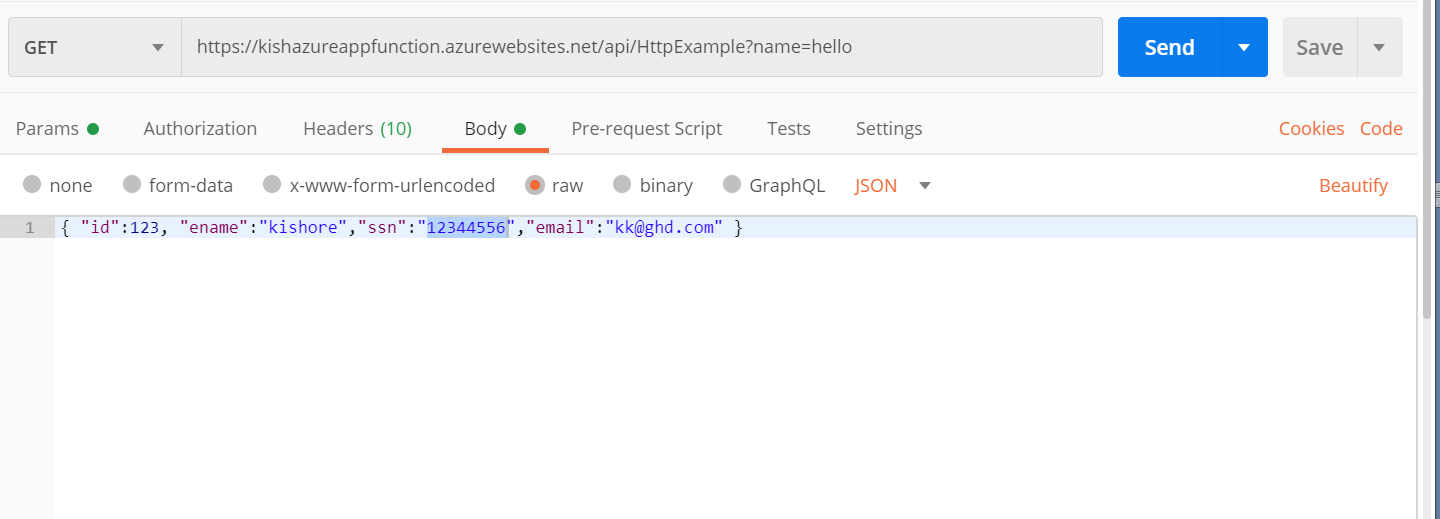
Click on the log console



8. Go to the postman add the new request with request parameter name as hello

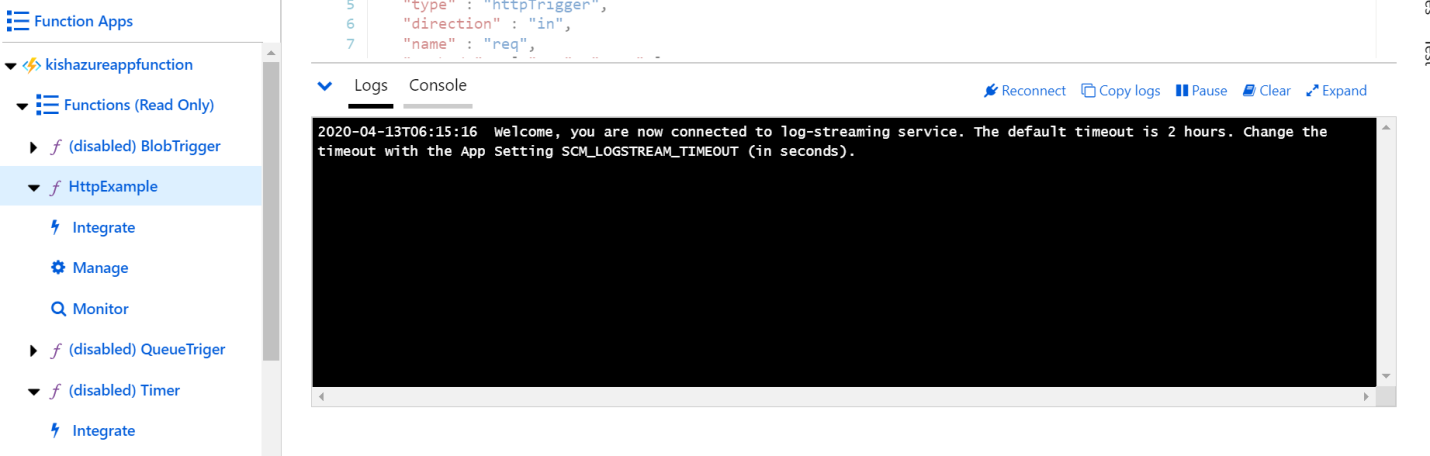


And request body simple employee json.

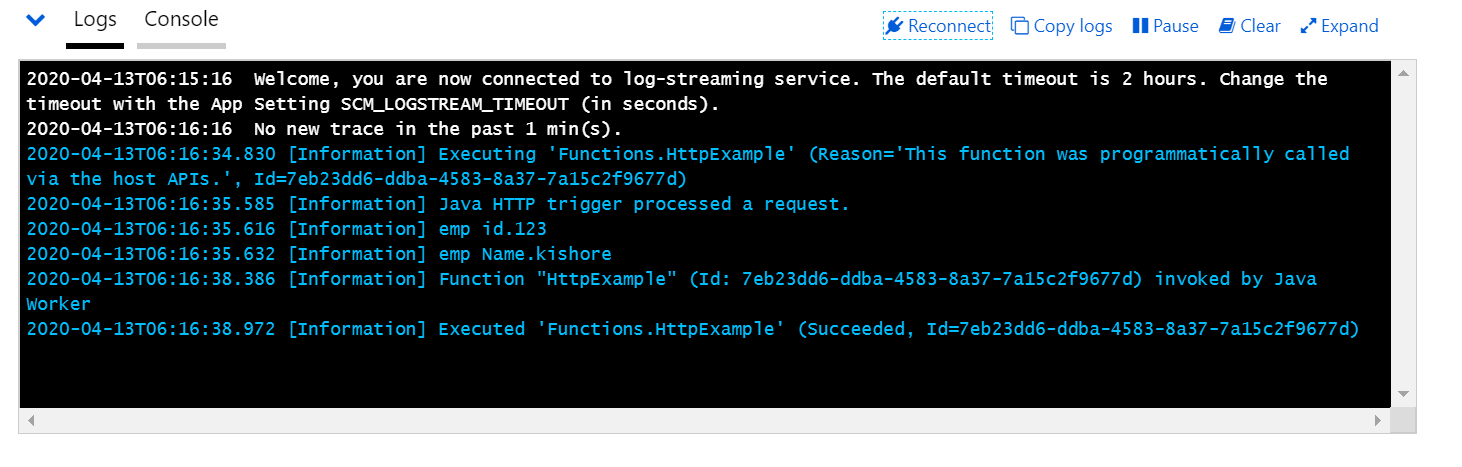


Click on send request

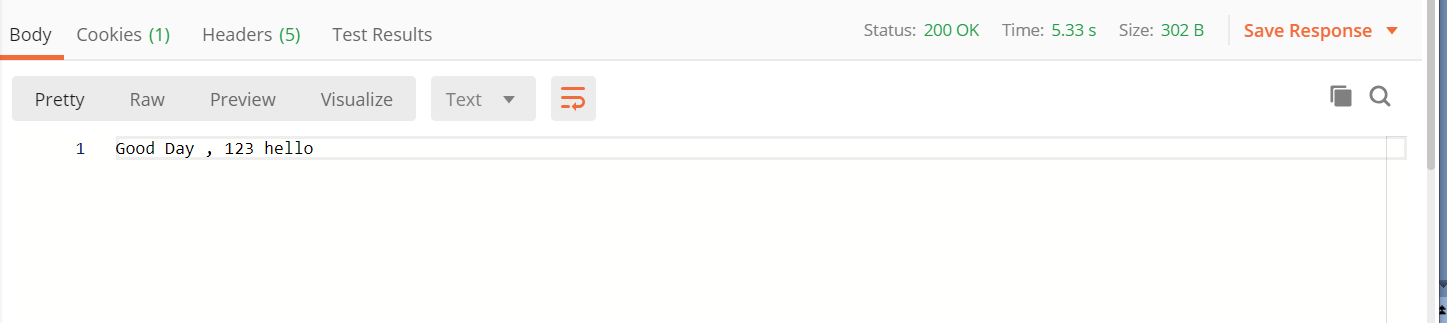
9. Navigate to the HTTPExample logs console on azure portal



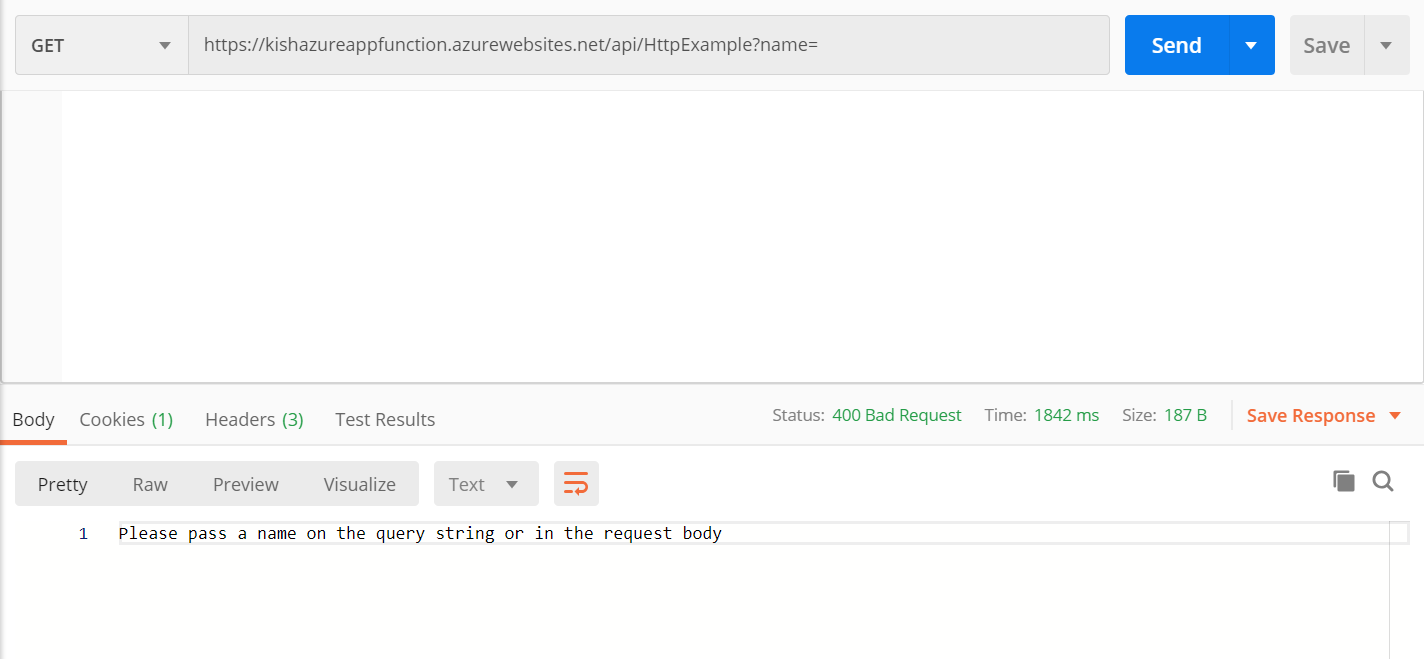
You can find the request parameter value and employee object details on logger console.



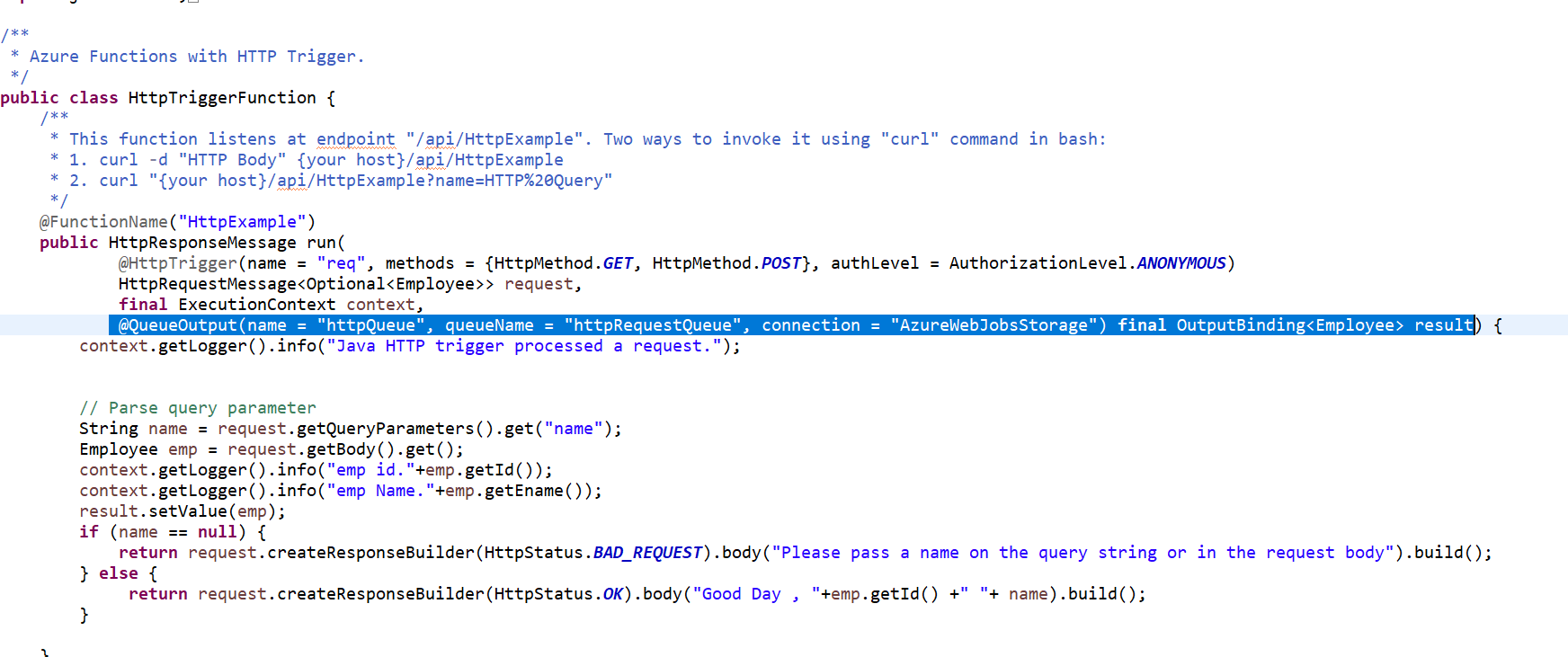
Http Response posted status with 200.



10. If request parameter value is null then Http Response posted status with 400



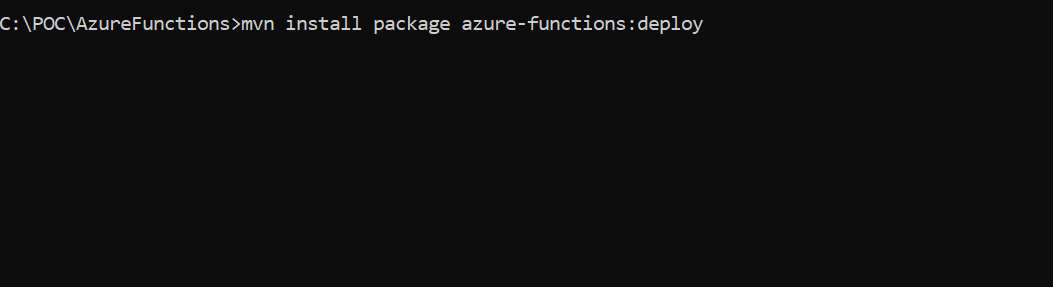
11.**To the above HTTP trigger code I have added new input parameter @QueueOutput so that Employee pojo would be written to azure storage queue.**





Deploy the Azure function use the following command

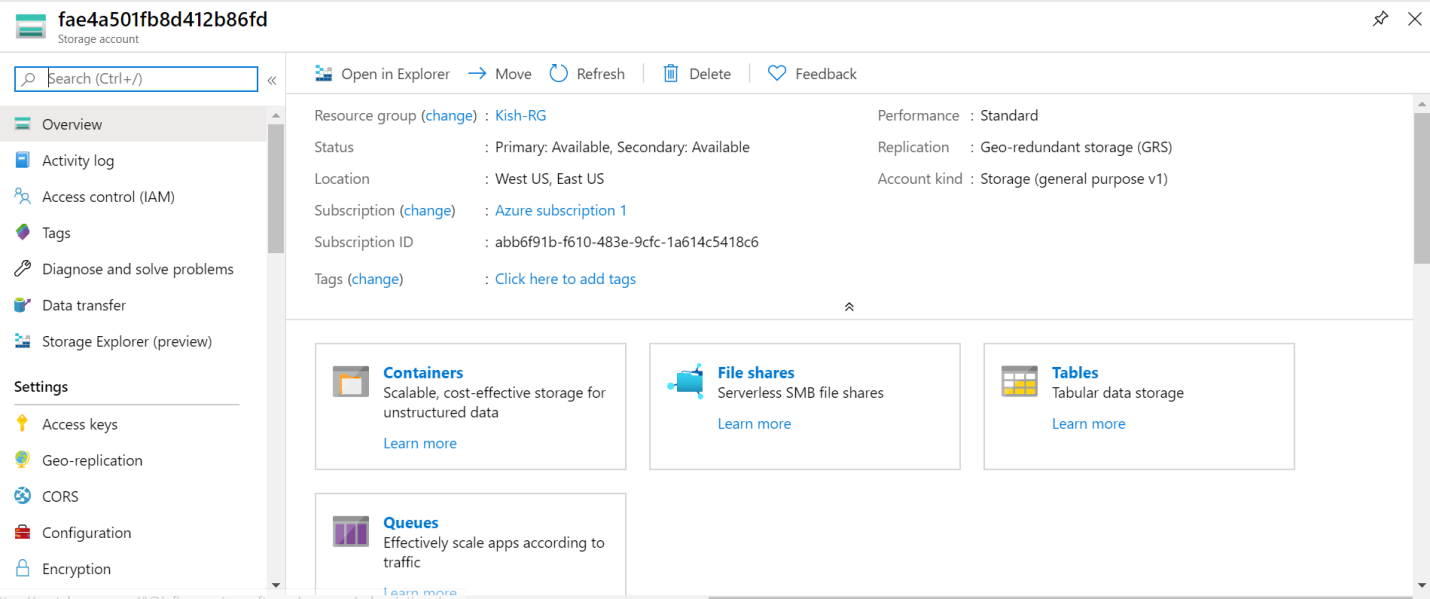
mvn install package azure-functions:deploy



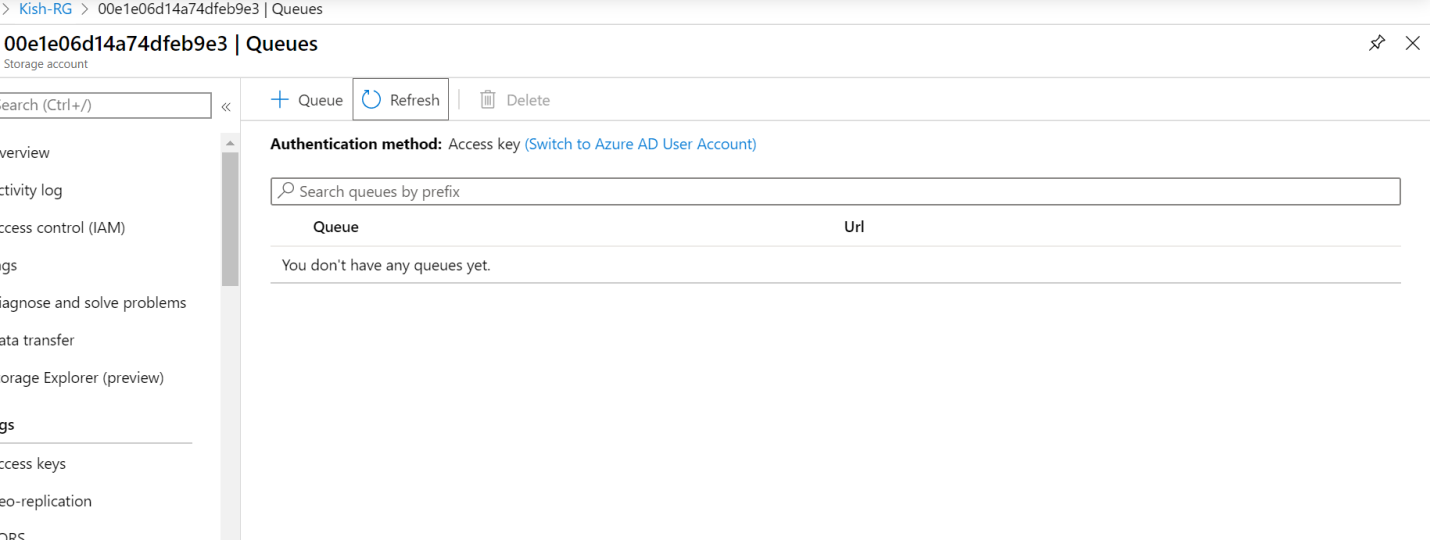
12. System automatically creates Create a new azure storage under resource group Kish-RG



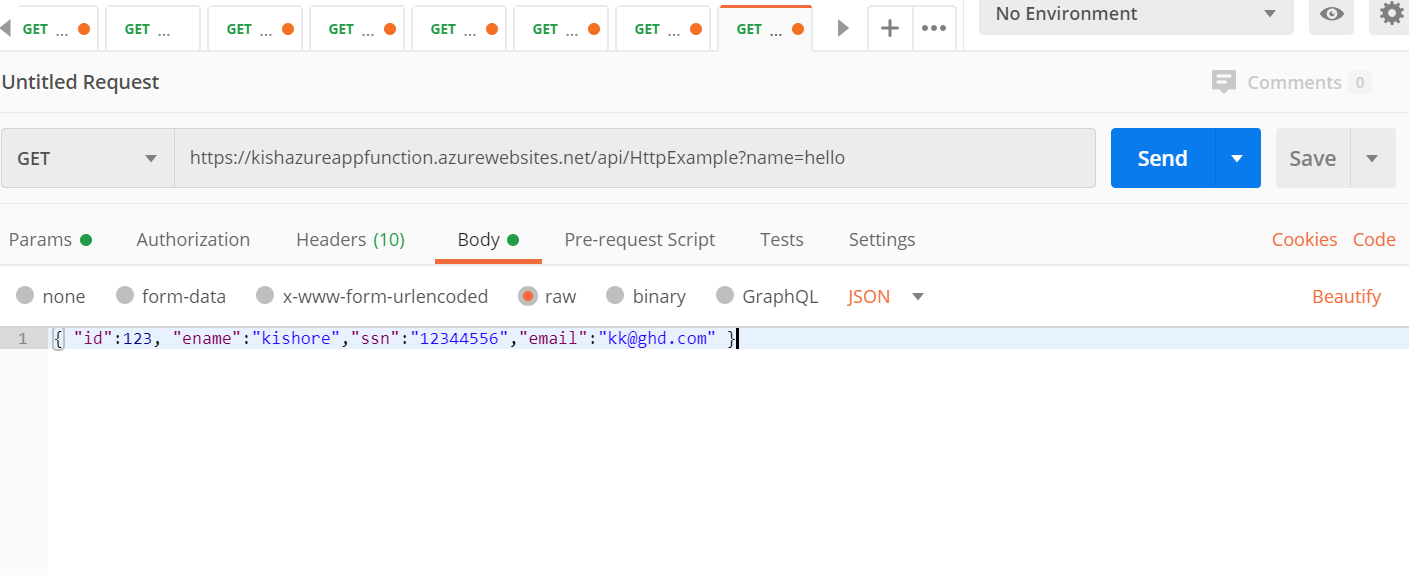
13.Navigate to the storage account -> Queues



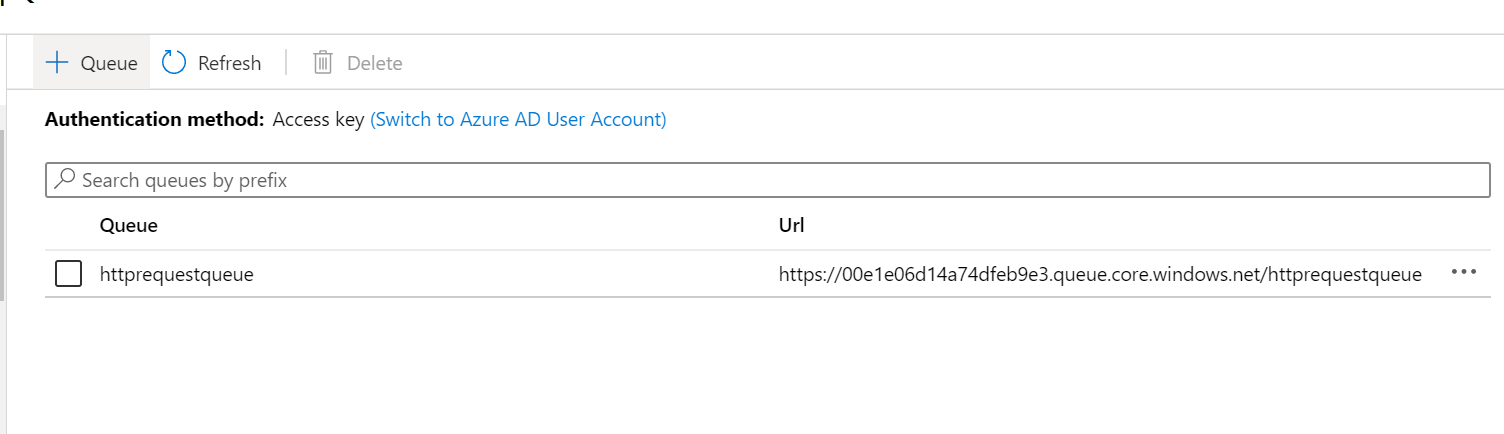
No queue found under Queuees



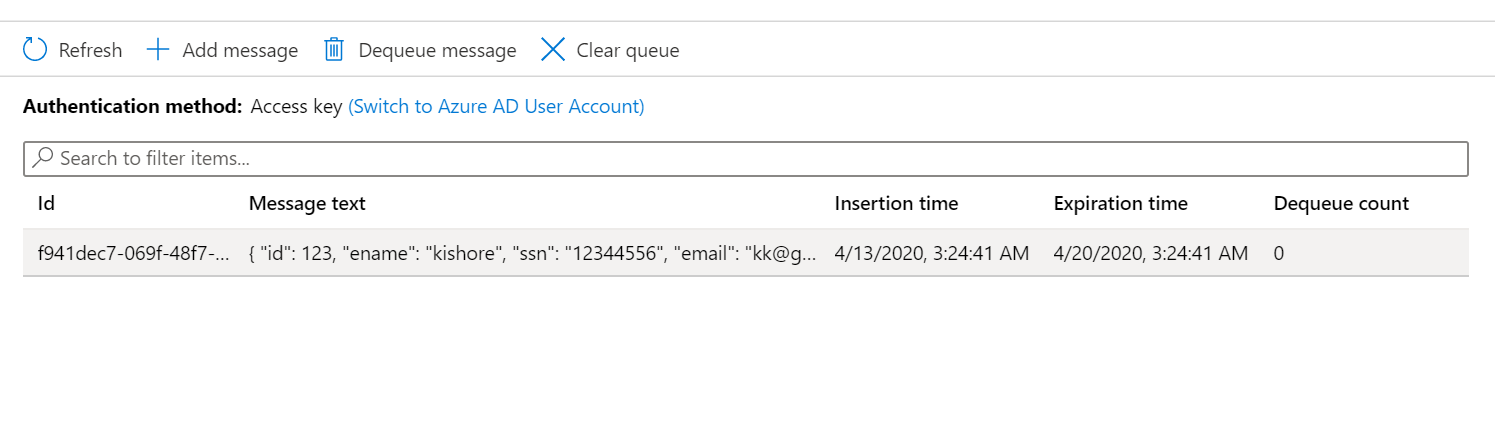
Send a new request to Http Trigger



New queue created with without human intervention

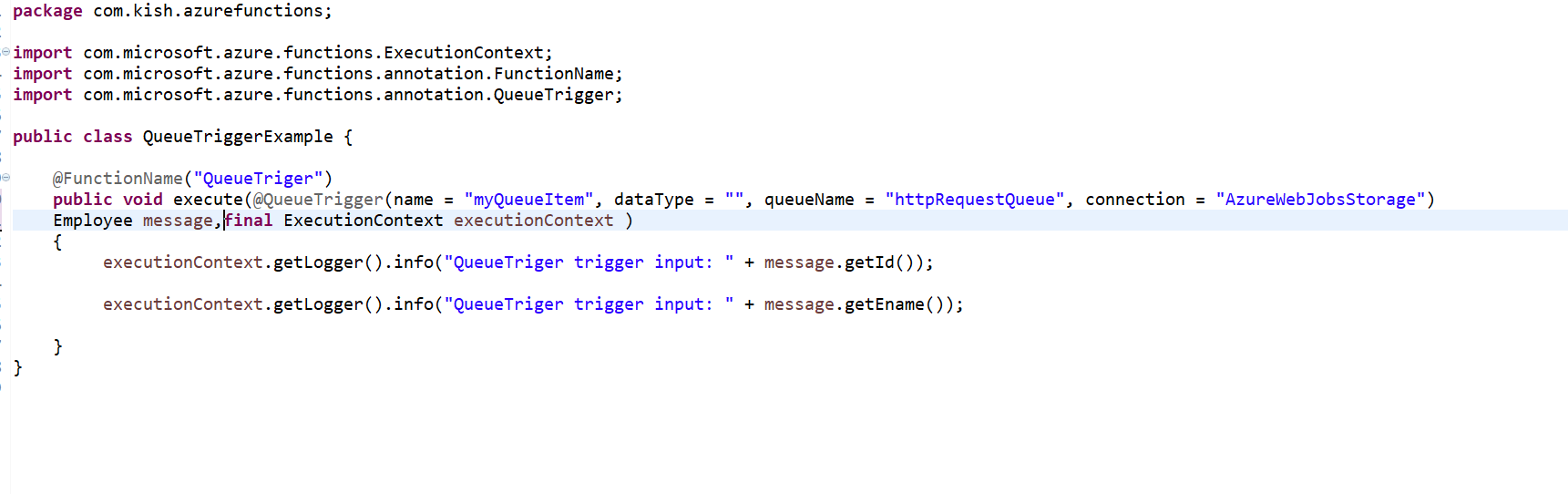


Queue contains message text as Employee JSON object .



**Queue Trigger**

14. Create the Queue Trigger Function which reads Employee POJO from httpRequestQueue (Message posted in previous HttpTrigger function).

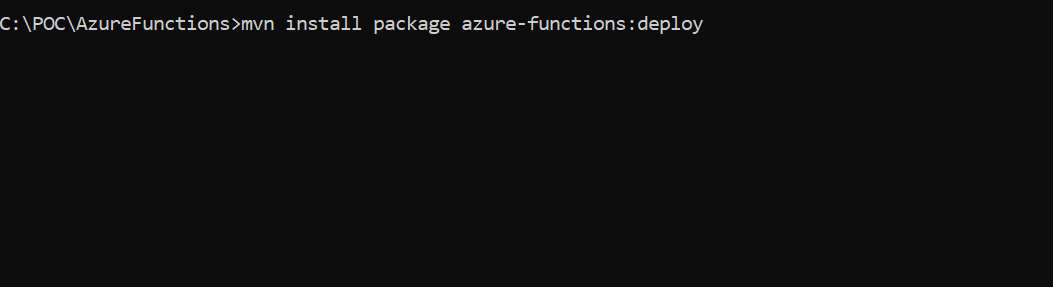


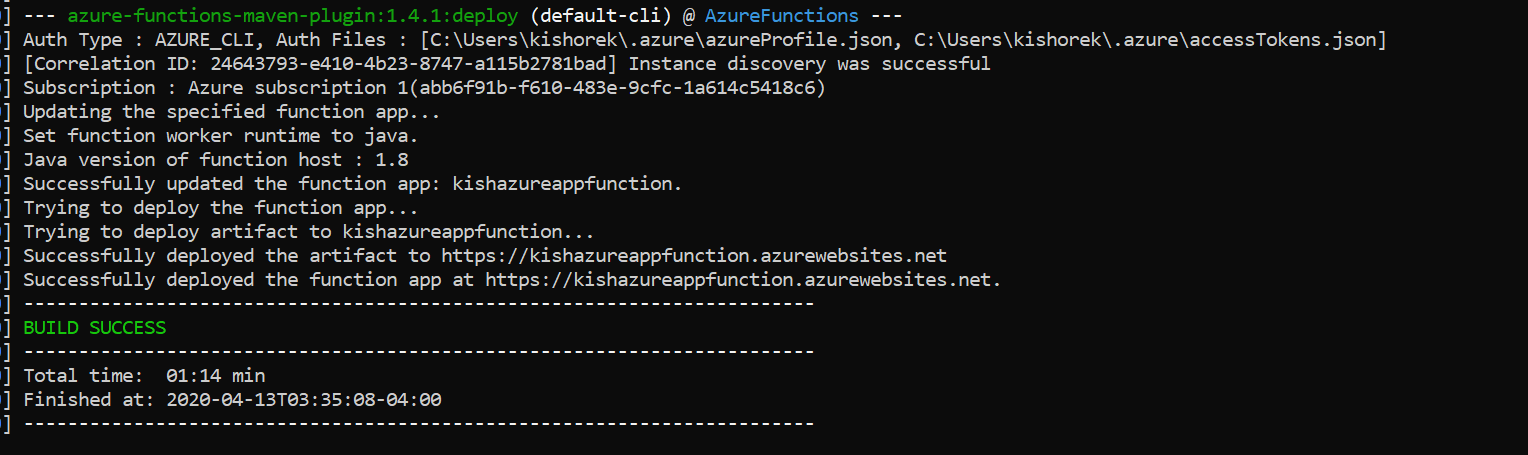
*@QueueTrigger(name = "myQueueItem",* ***dataType = "",*** *queueName = "httpRequestQueue", connection = "AzureWebJobsStorage") Employee message*

*Data type attribute should be empty so that message text will be converted to Employee pojo*

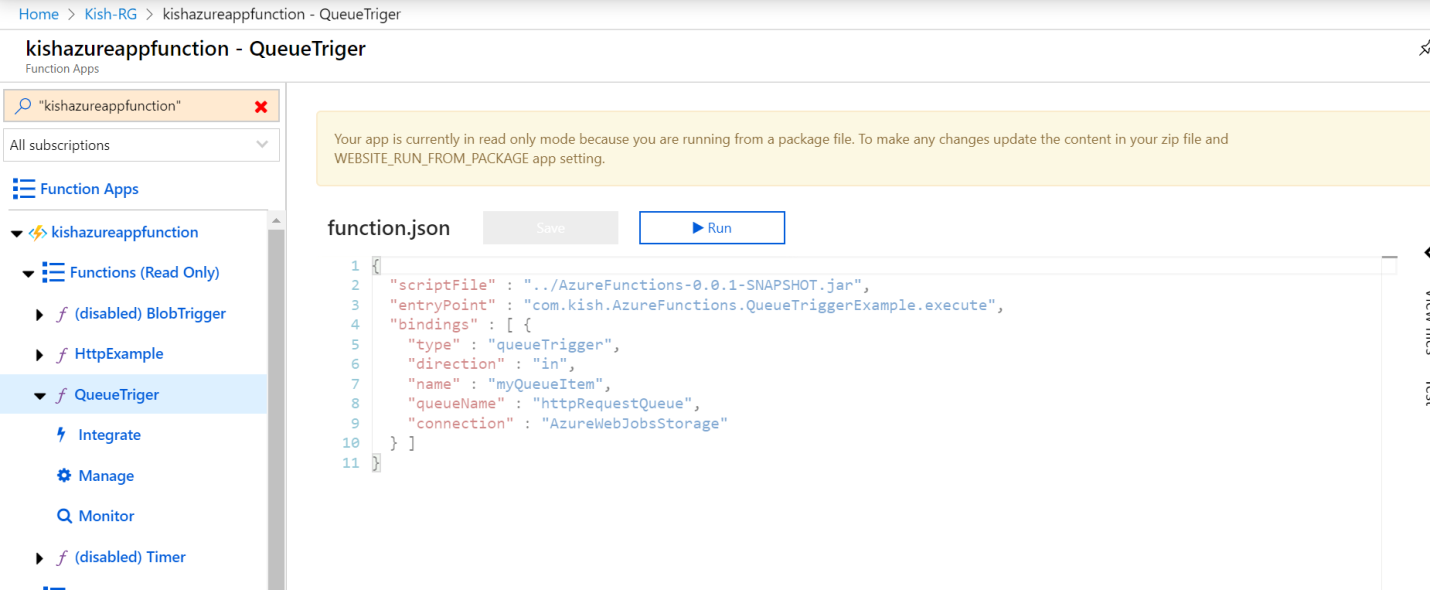
Deploy the Azure function use the following command

mvn install package azure-functions:deploy

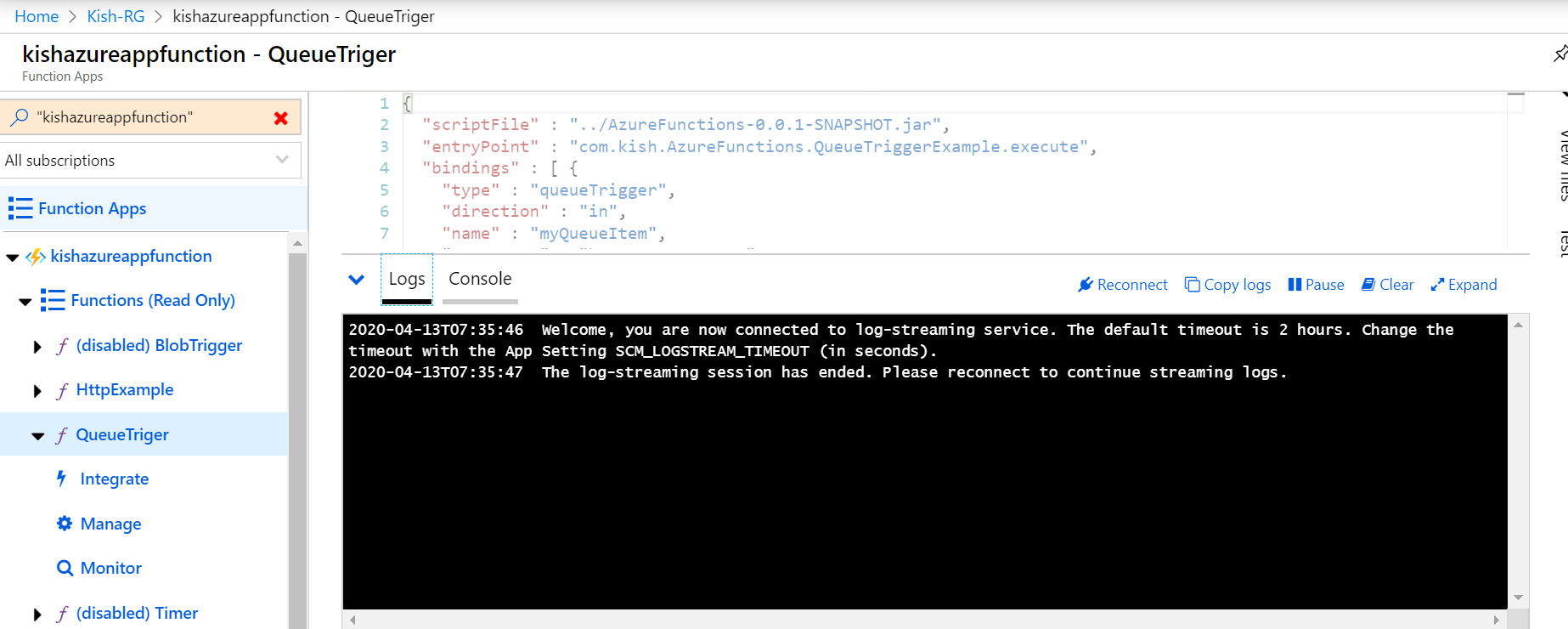




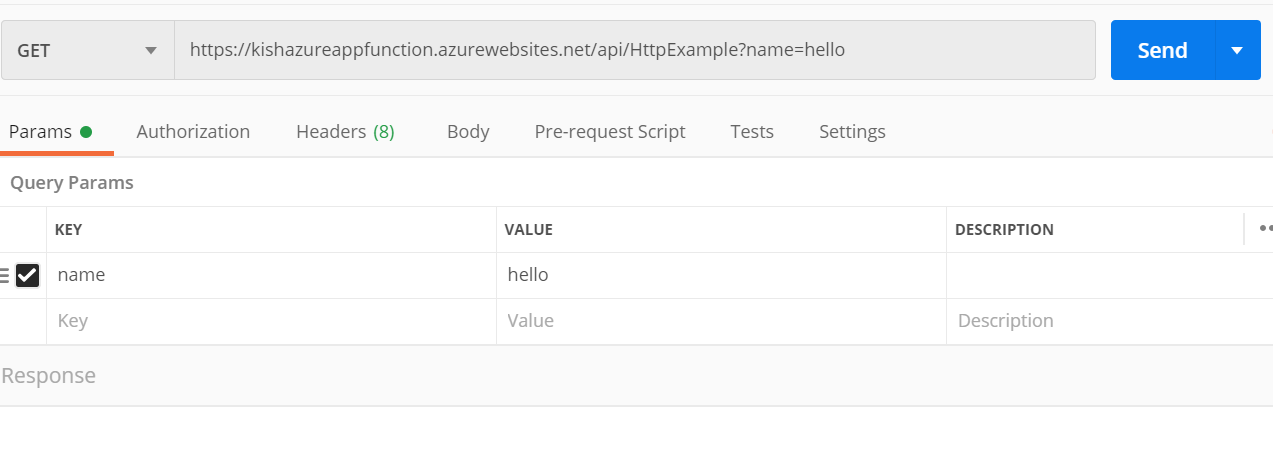
15. Navigate to the appfunction



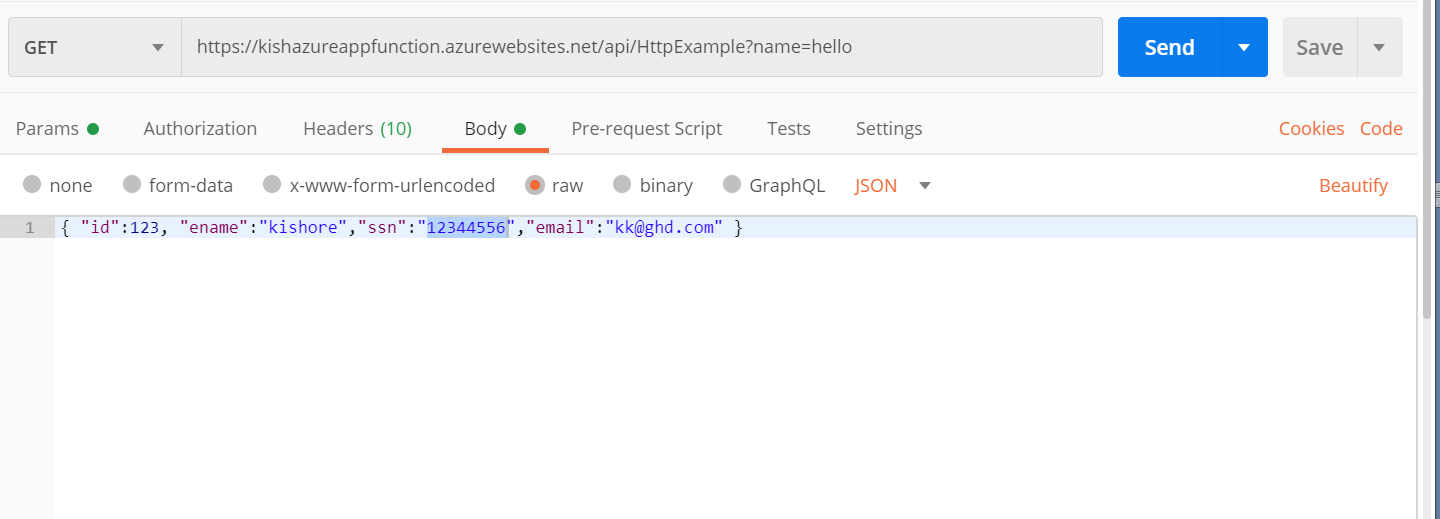
16.Navigate to the QueueTriger logs console on azure portal



17. Go to the postman add the new request with request parameter name as hello

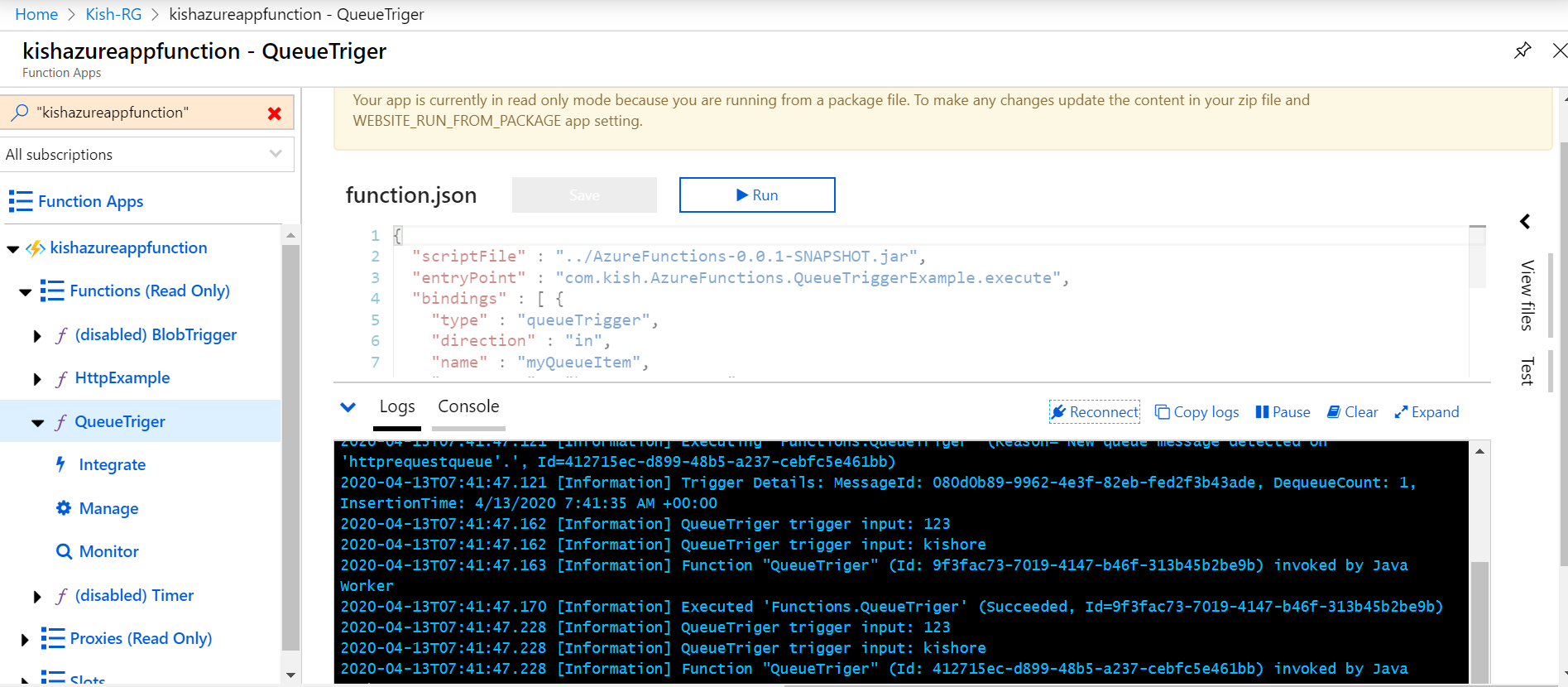


And request body simple employee json.



Click on send request

18. Employee object details printed on logger console.

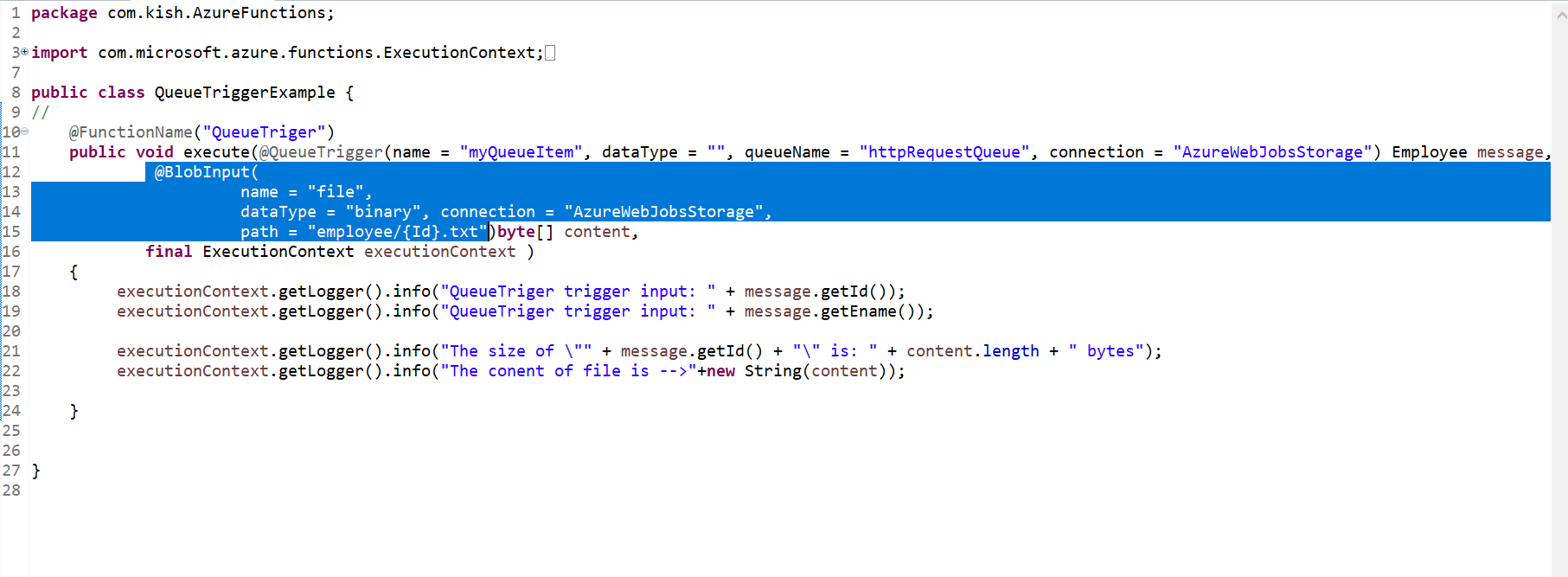


20.**To the above QueueTrigger code I have added new input parameter @BlobInput so that it will read a blob file from azure storage Blob conatiner based on filename matches with Employee🡪Id.**

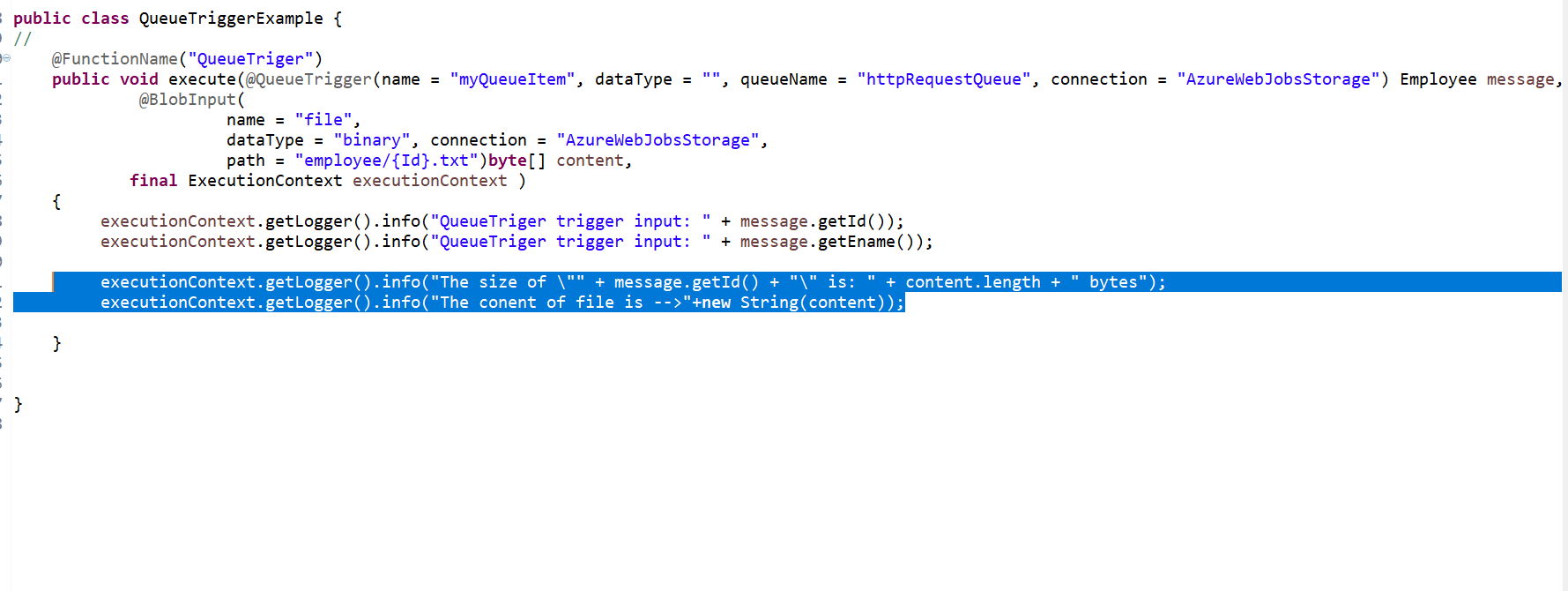
@BlobInput(name = "file", dataType = "binary", connection = "AzureWebJobsStorage",

path = "employee/{Id}.txt")**byte**[] content

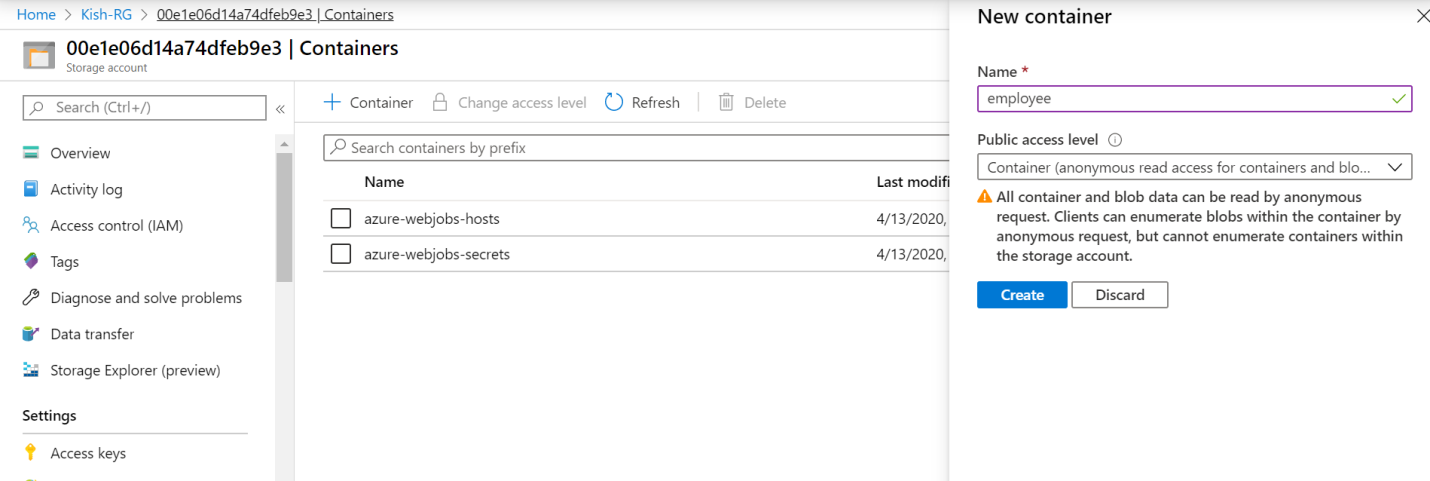
The {Id} value is Employee Object 🡪id property



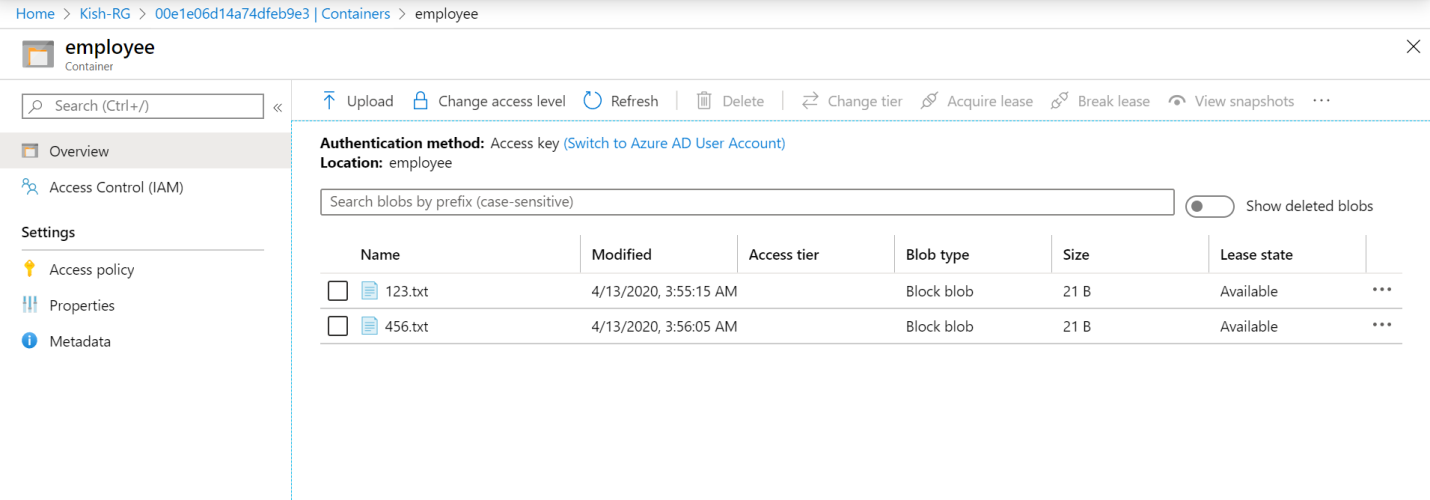
**Print the content size and content in log console.**

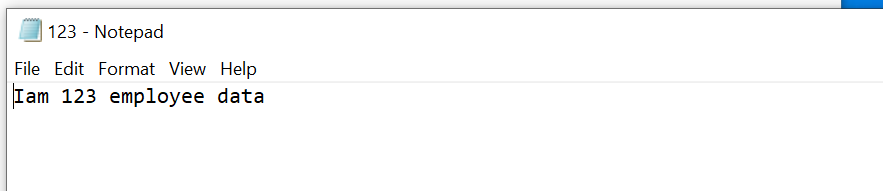


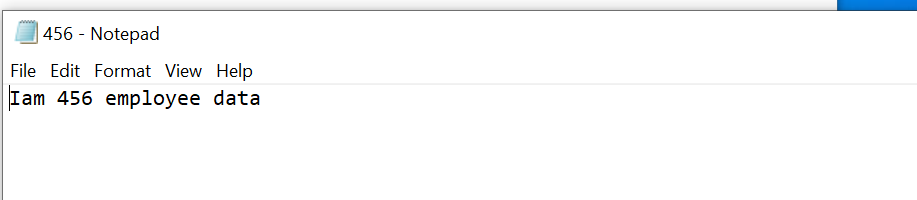
21. Create a new blob conatiner in storage account



22.I have uploaded two text files to employee conatiner

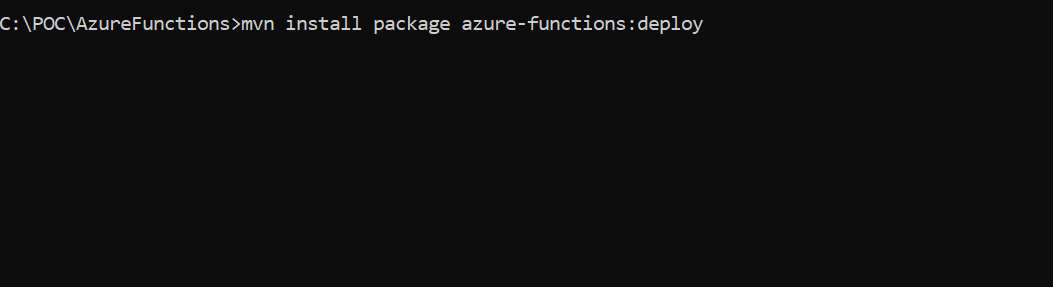


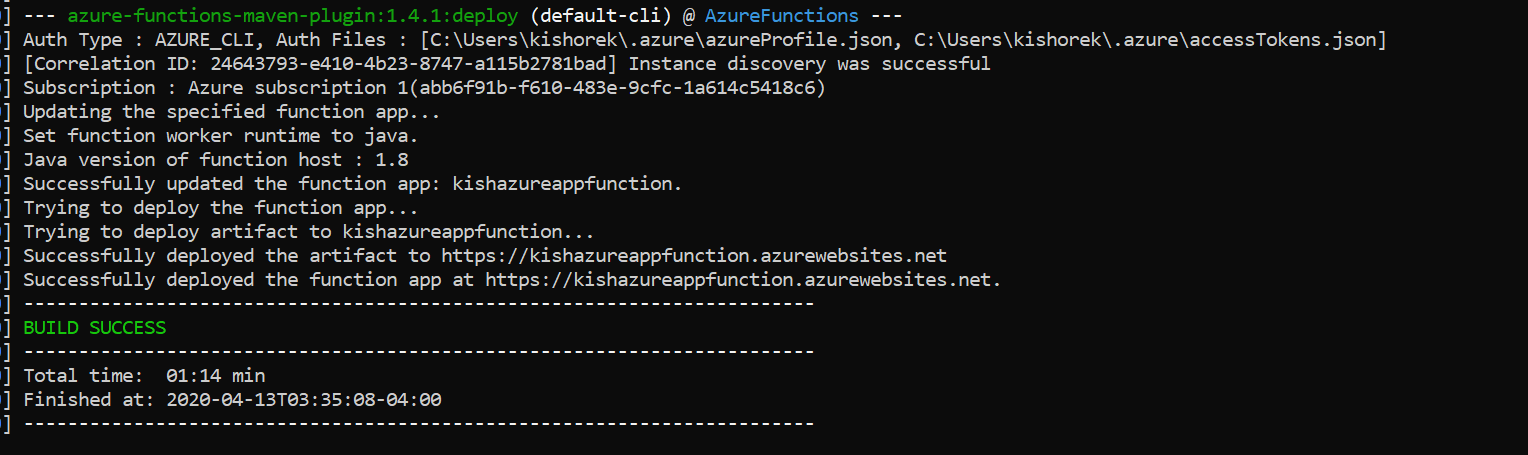




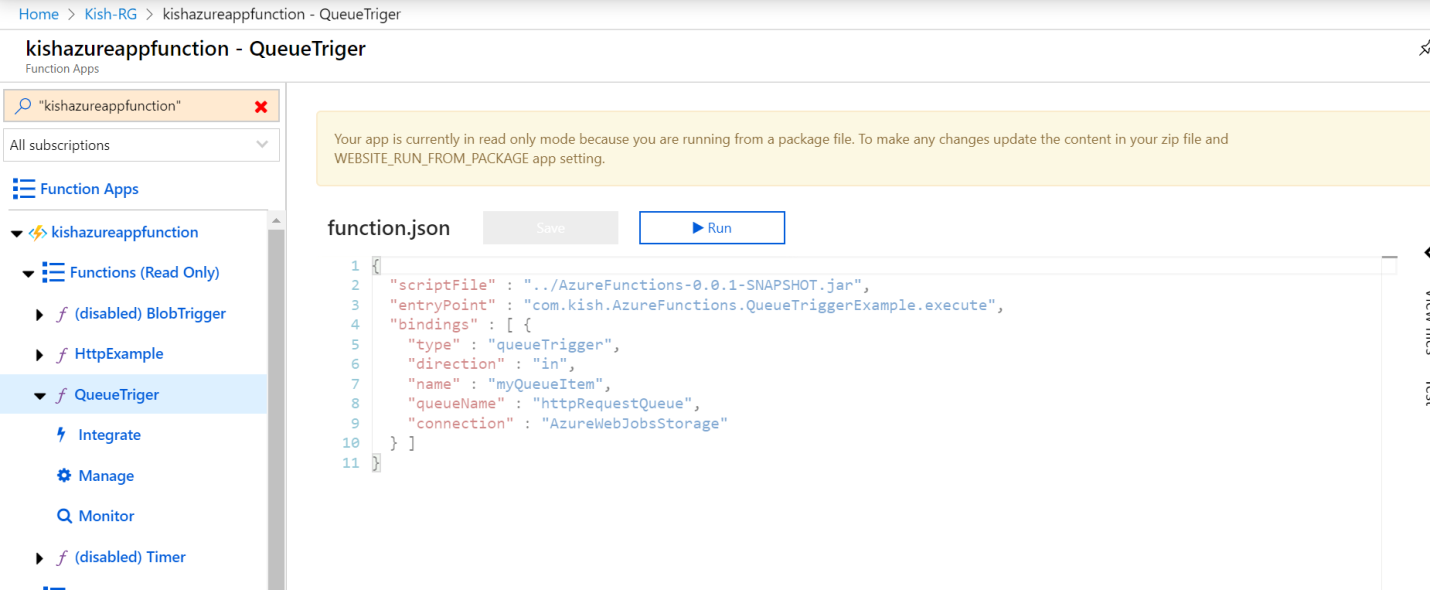
Deploy the Azure function use the following command

mvn install package azure-functions:deploy

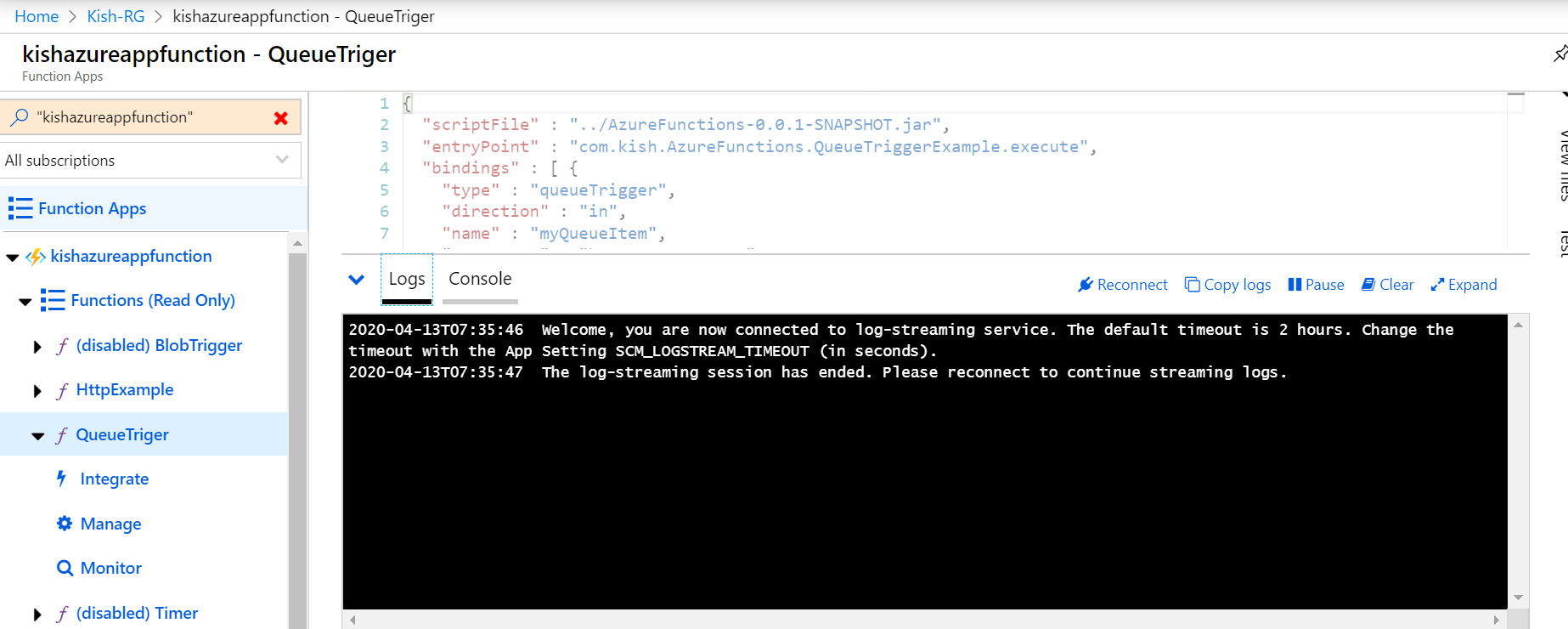




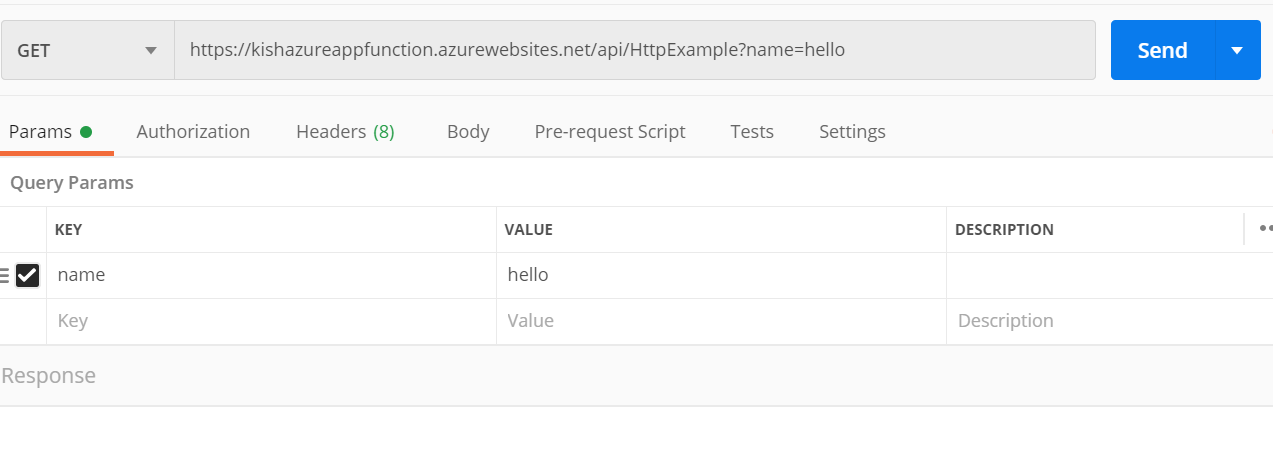
23. Navigate to the appfunction



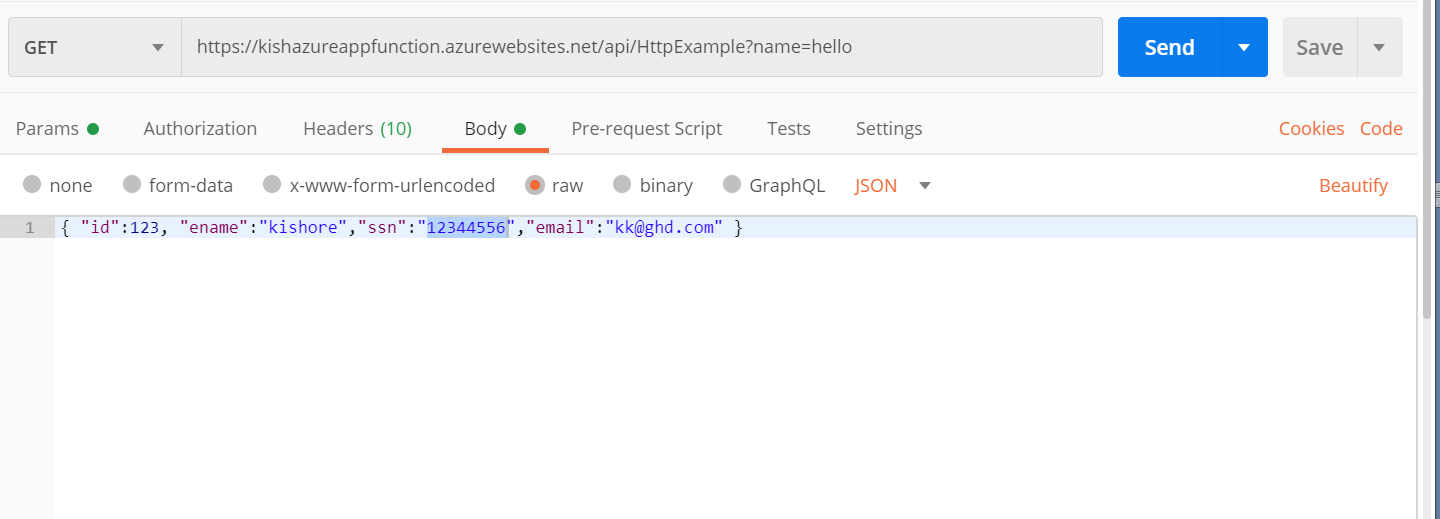
24. Navigate to the QueueTriger logs console on azure portal



25. Go to the postman add the new request with request parameter name as hello

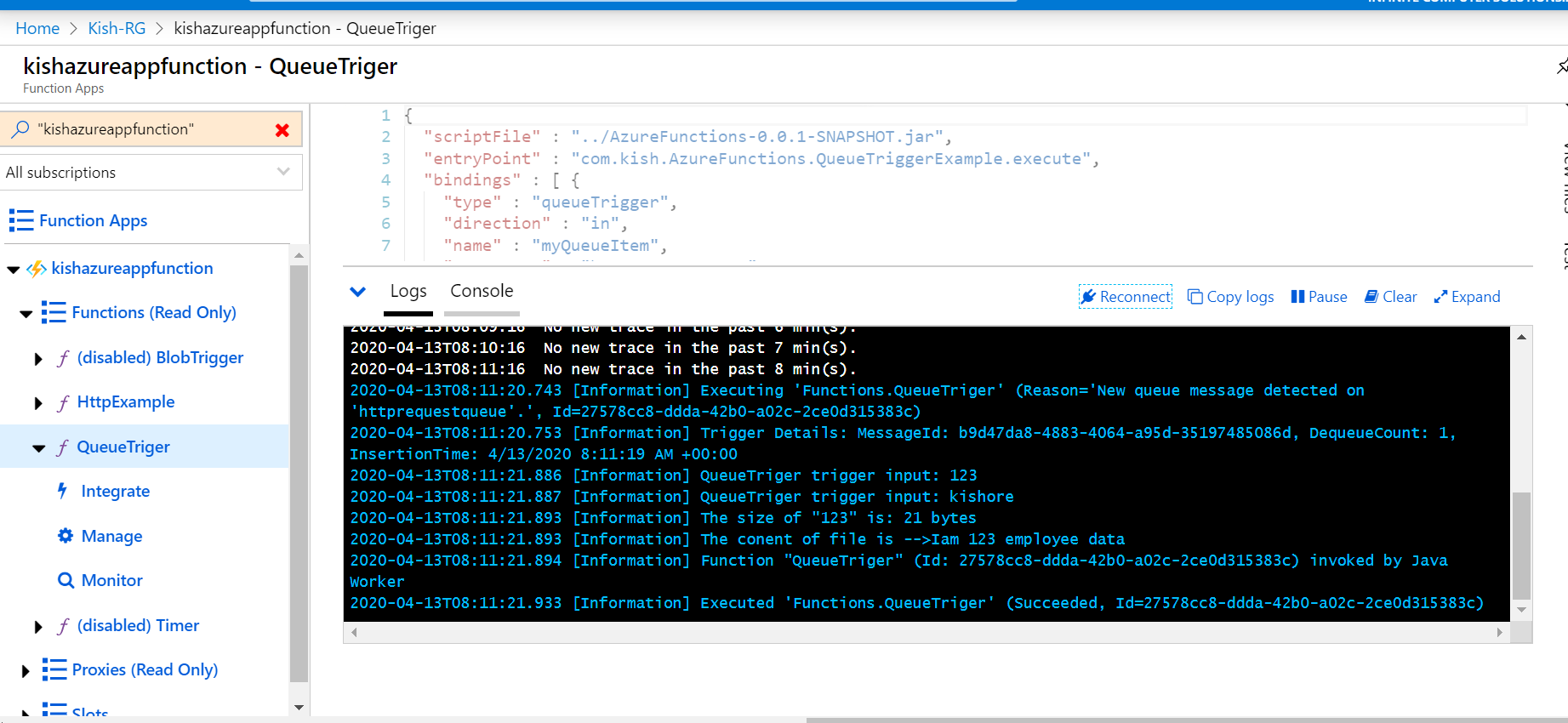


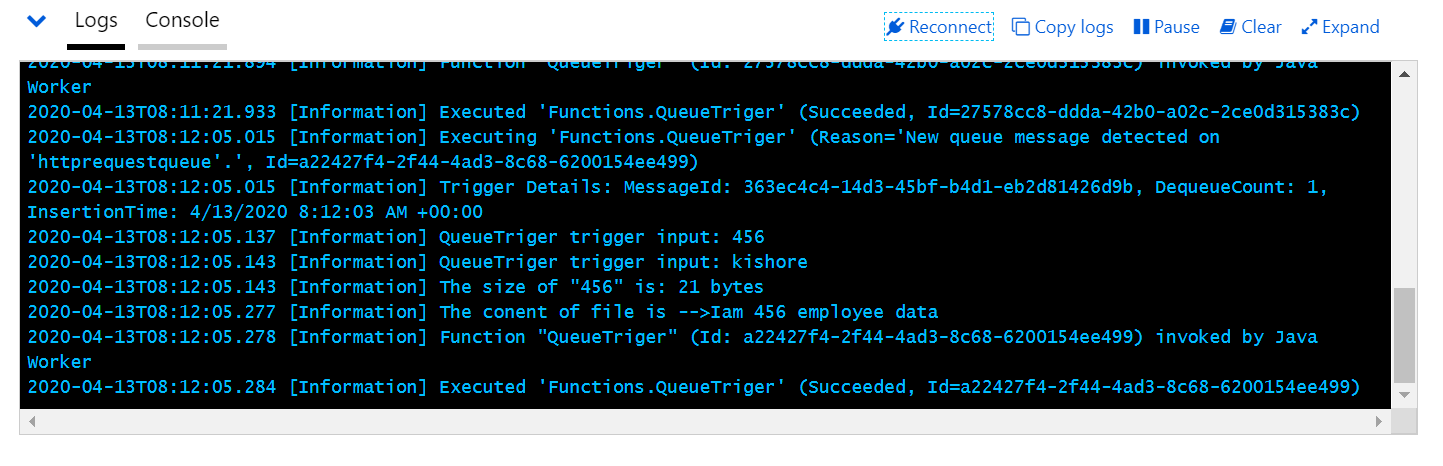
And request body simple employee json.



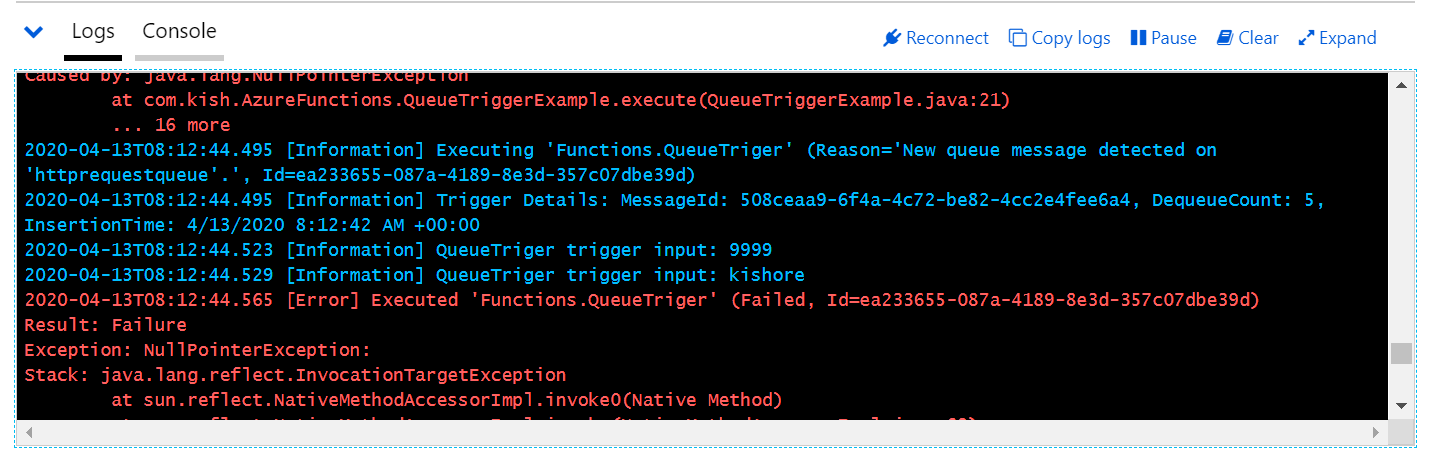
Click on send request

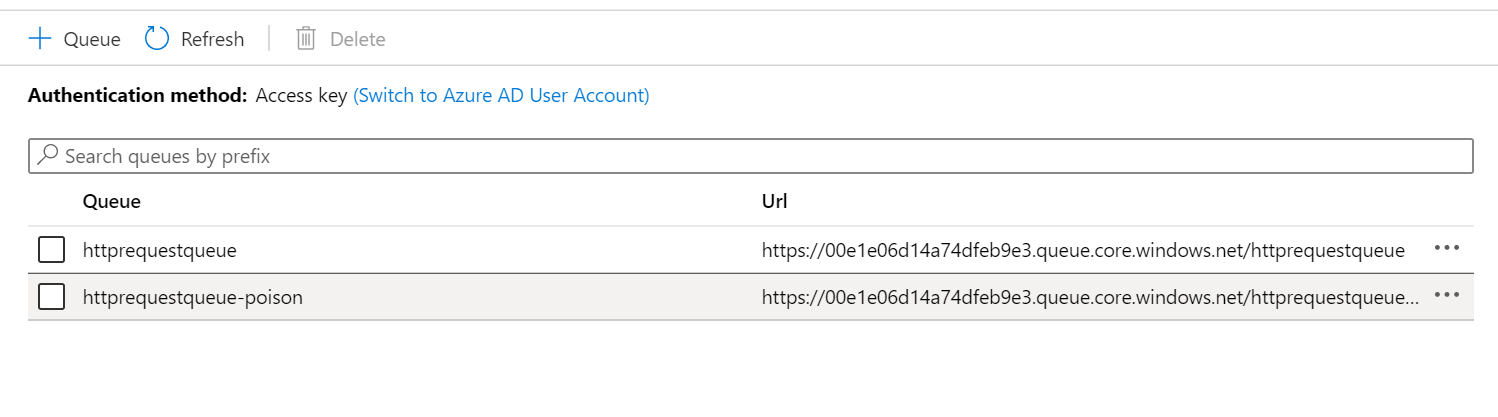
26. Base on employee id blob file details printed on logger console.

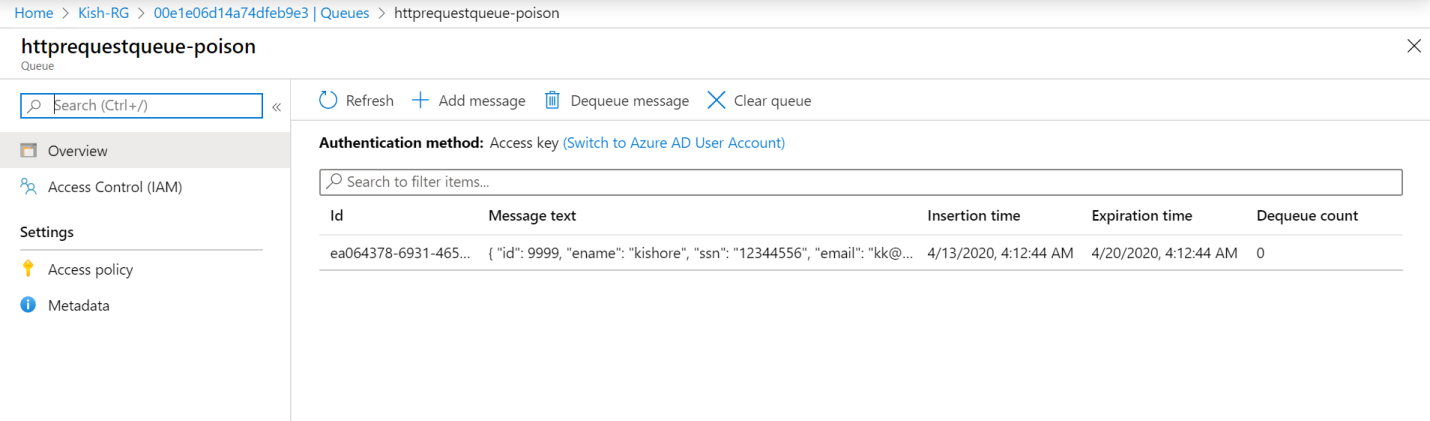




With employee id blob file is not found in the container system throws exception and system creates poison queue message placed into the poison queue

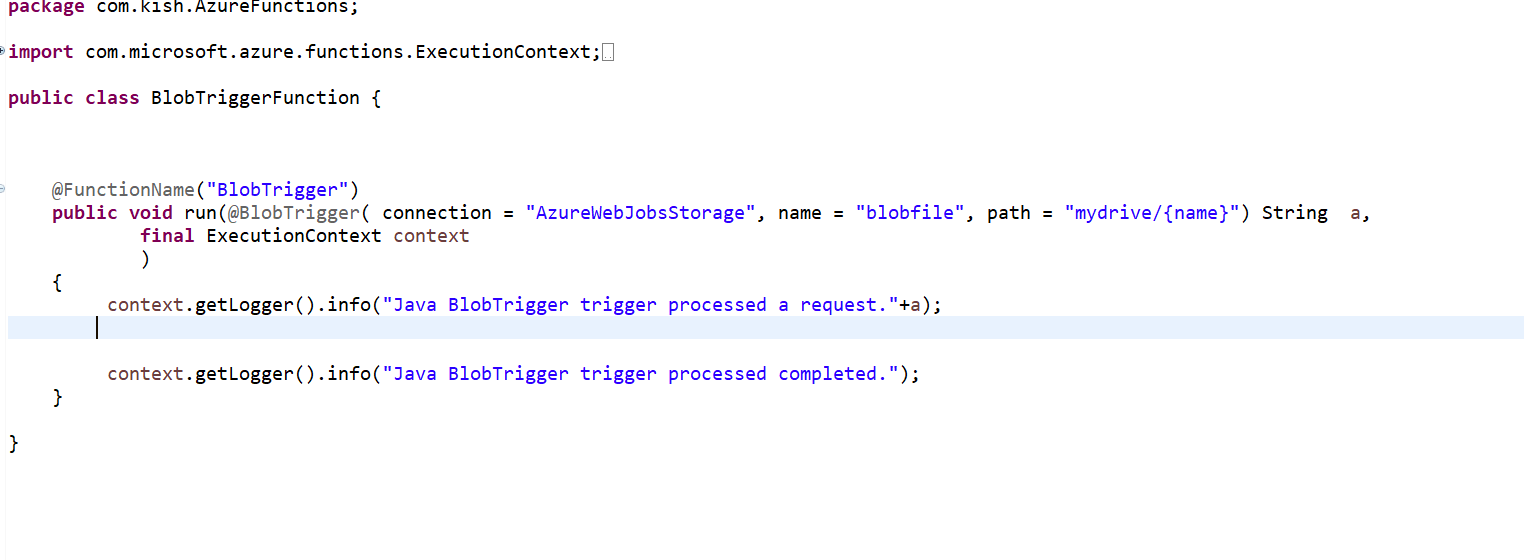




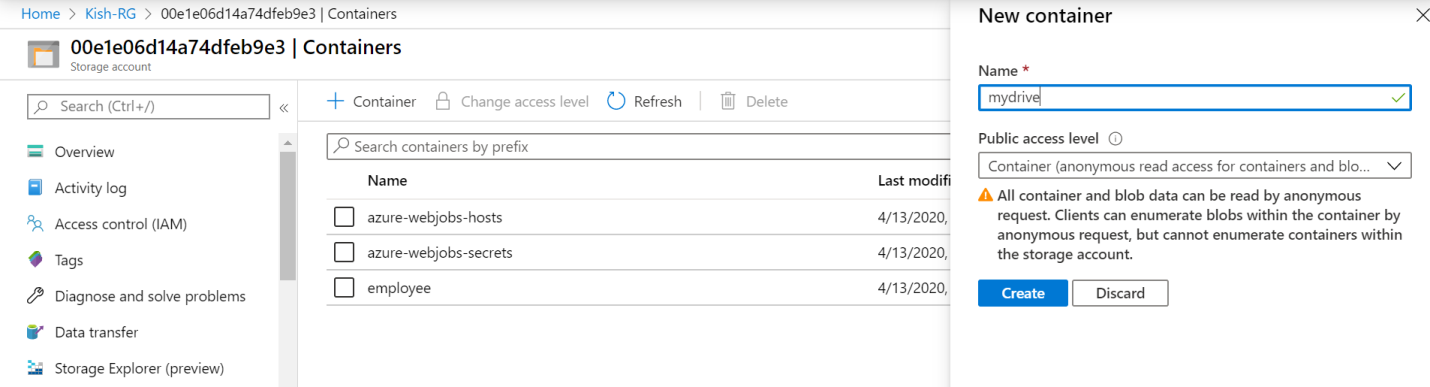


**Blob Trigger**

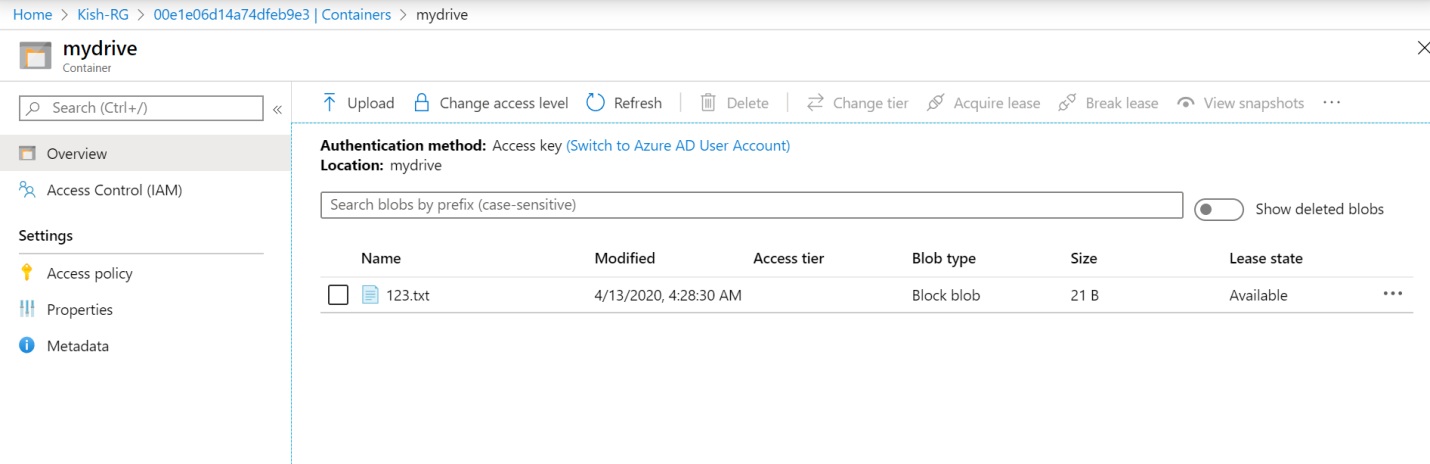
27. Create the Blob Trigger Function which logs the filename and size when a blob is addedor updated in the mydrive container



28.Creat mydrive container in Azure storage

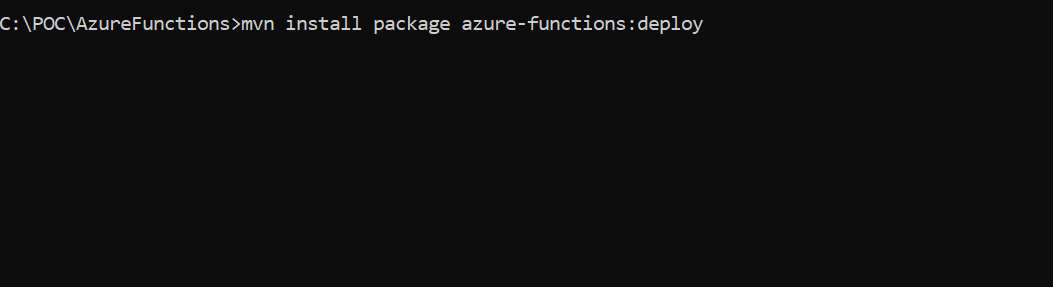


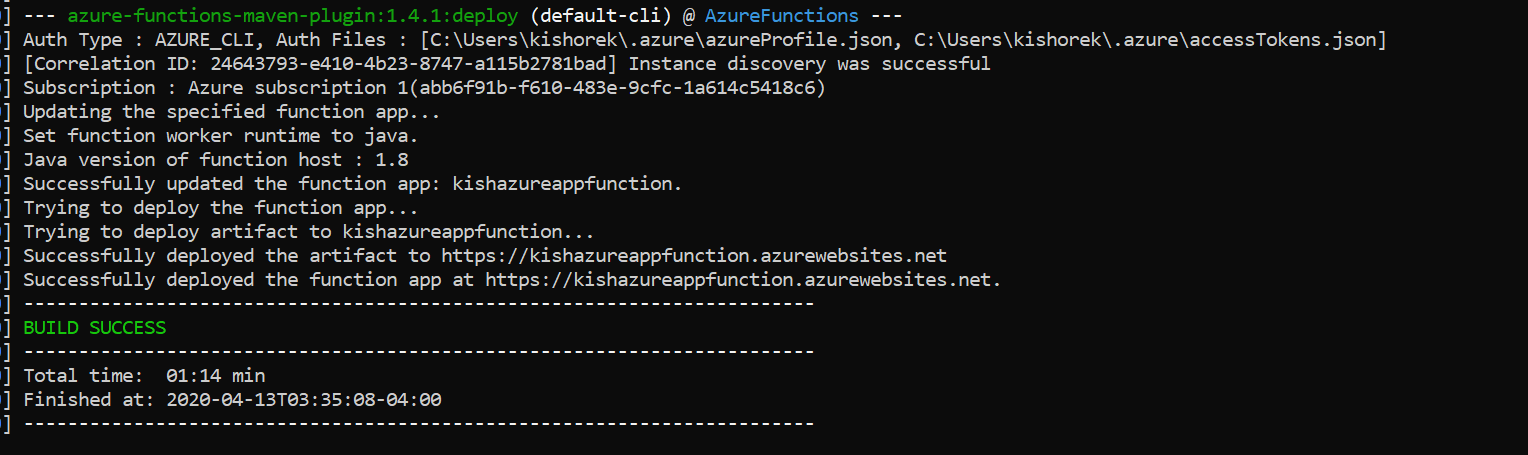
Upload sample file



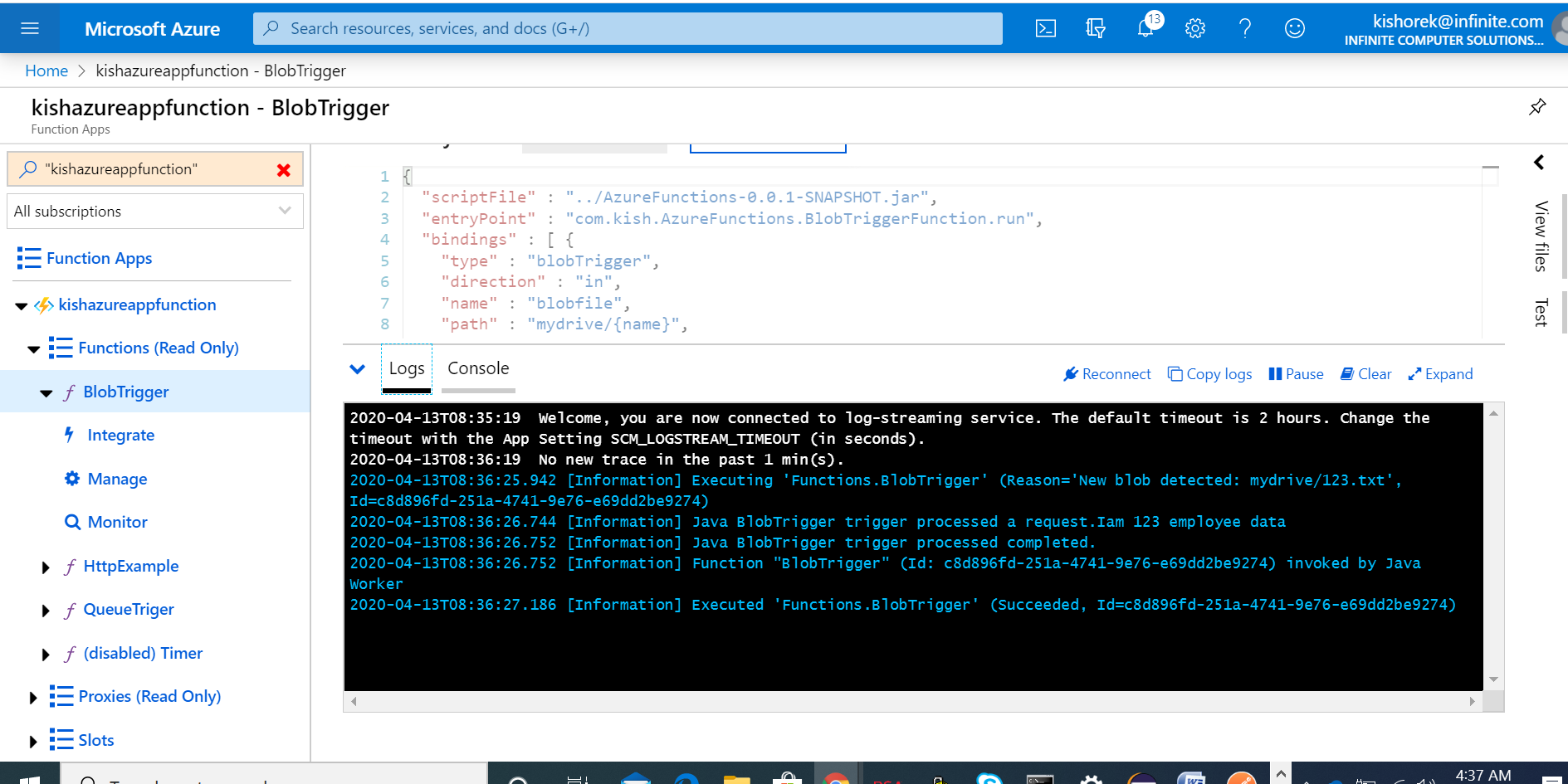
Deploy the Azure function use the following command

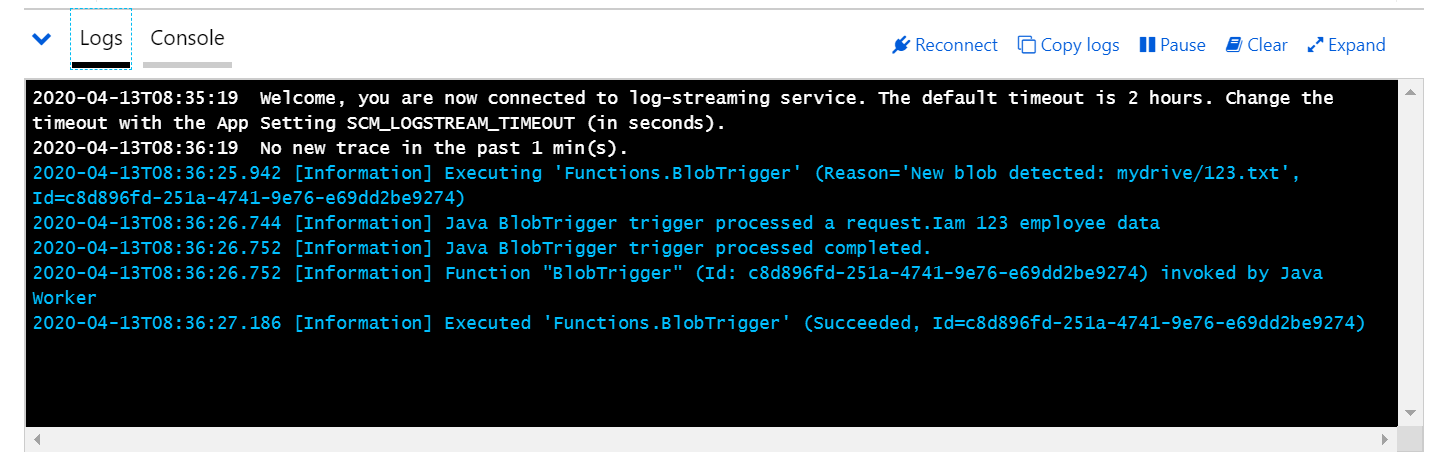
mvn install package azure-functions:deploy



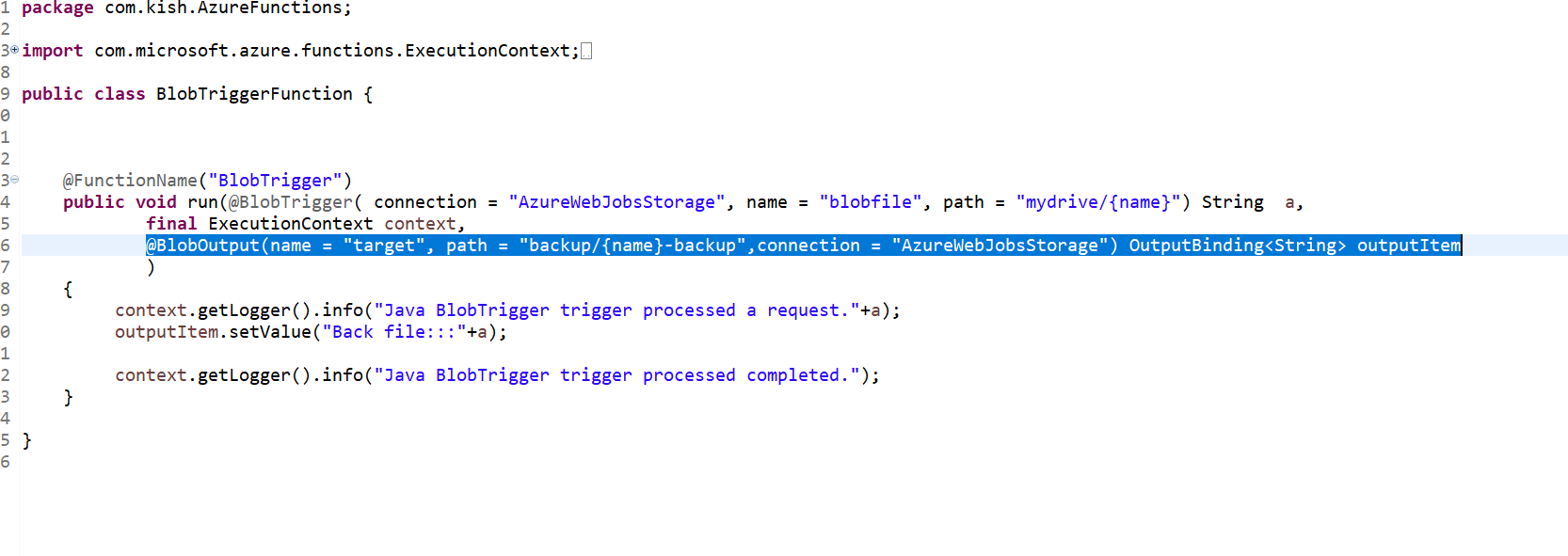


29. Navigate to the BlobTriger logs console on azure portal



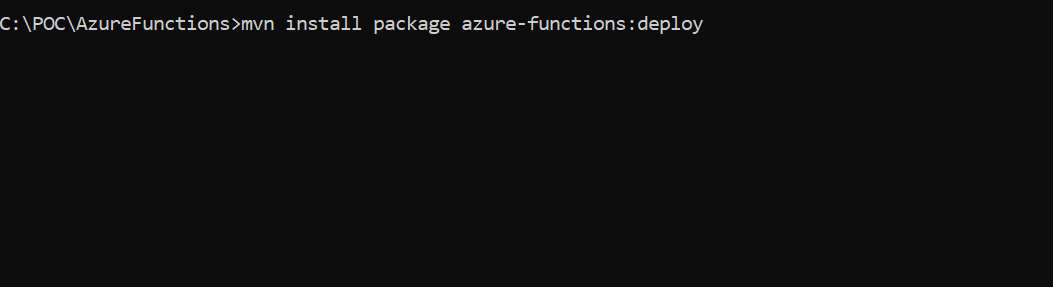


30.**To the above BlobTrigger code I have added new input parameter @BlobOutput so that it will read a blob file from azure storage Blob conatiner and makes a copy of a text blob.**

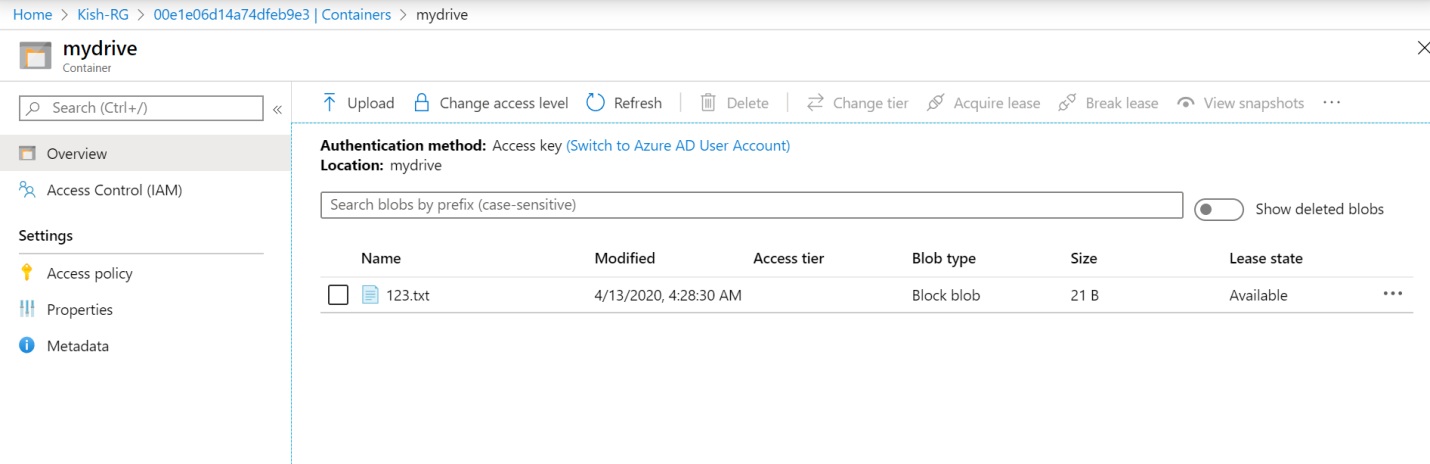
****

Deploy the Azure function use the following command

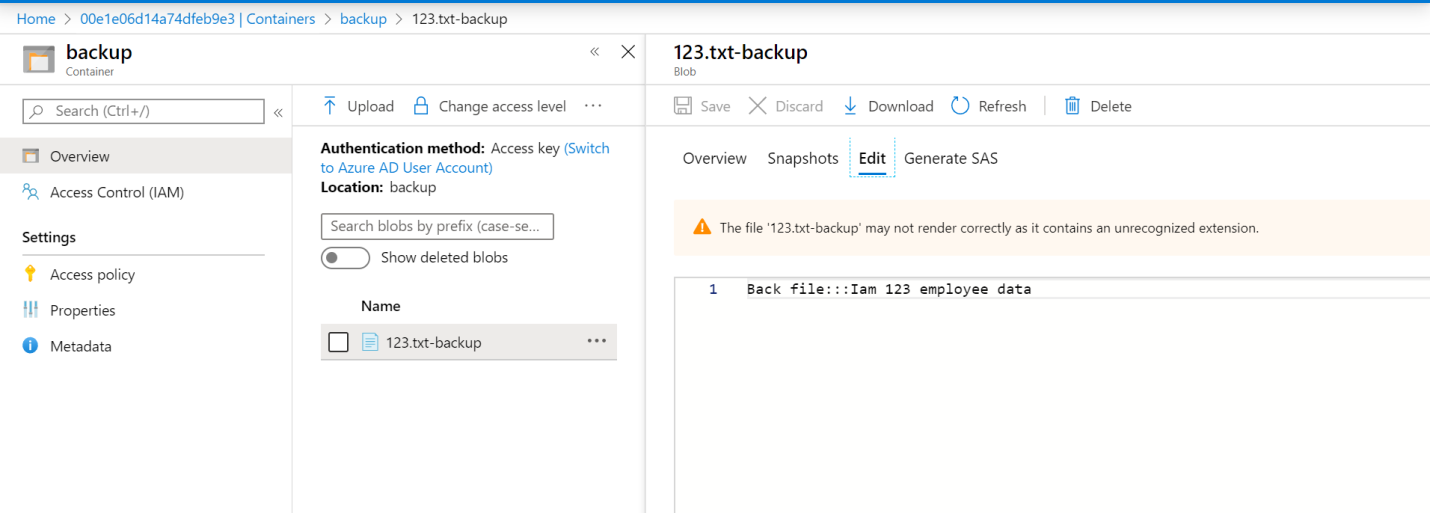
mvn install package azure-functions:deploy



Upload sample file

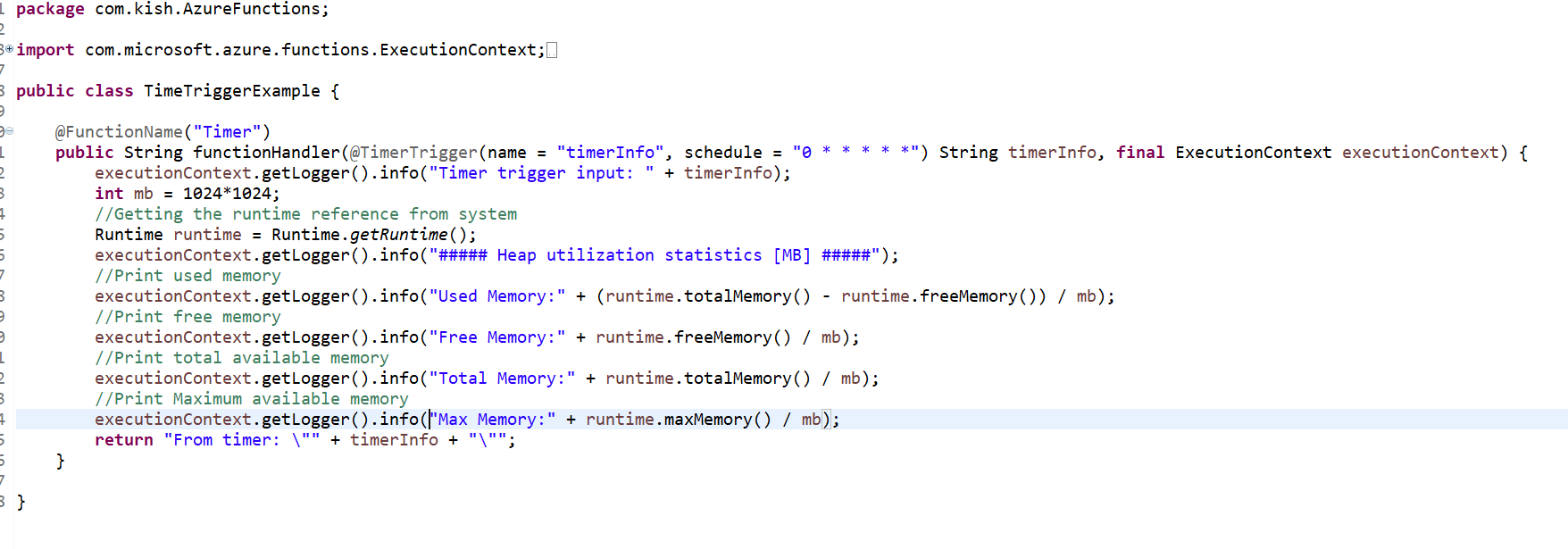


The backup container created with backup file



**TimeTrigger**

31. Create the Time Trigger Function runs based CRON expression schedule.The following function executes for every 1 minute

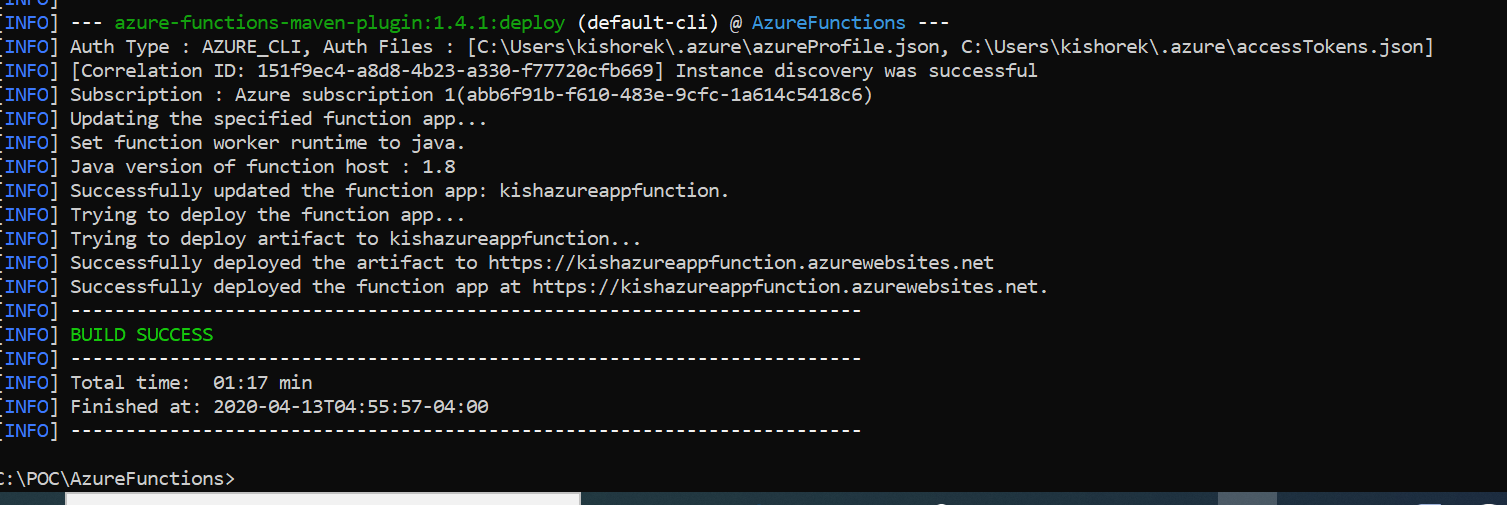


Use the following link to generate cron expression scheduler

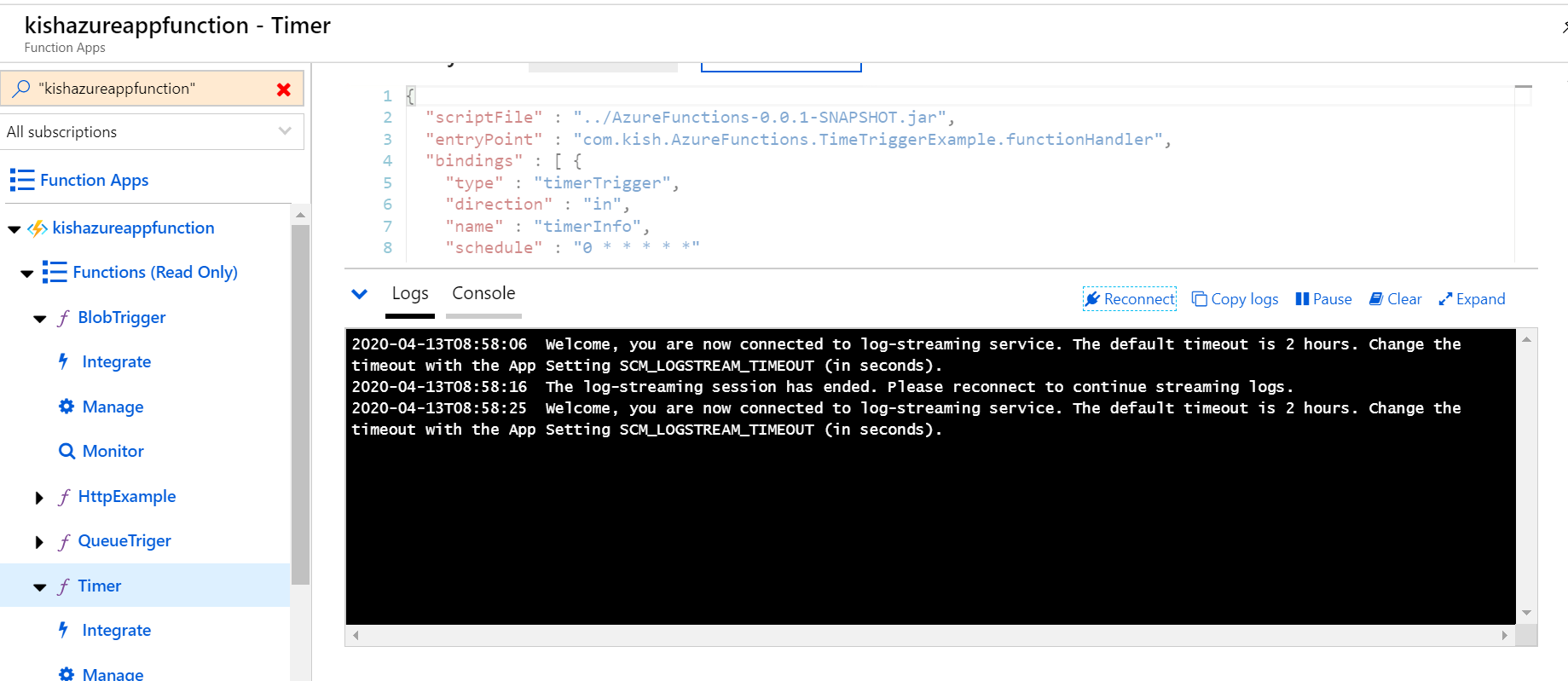
<https://www.freeformatter.com/cron-expression-generator-quartz.html>

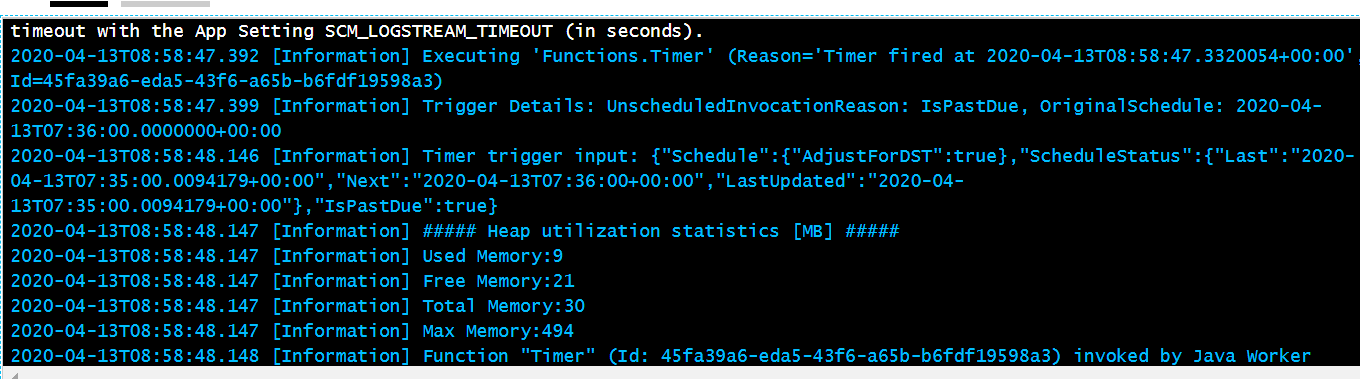
Deploy the Azure function use the following command

mvn install package azure-functions:deploy



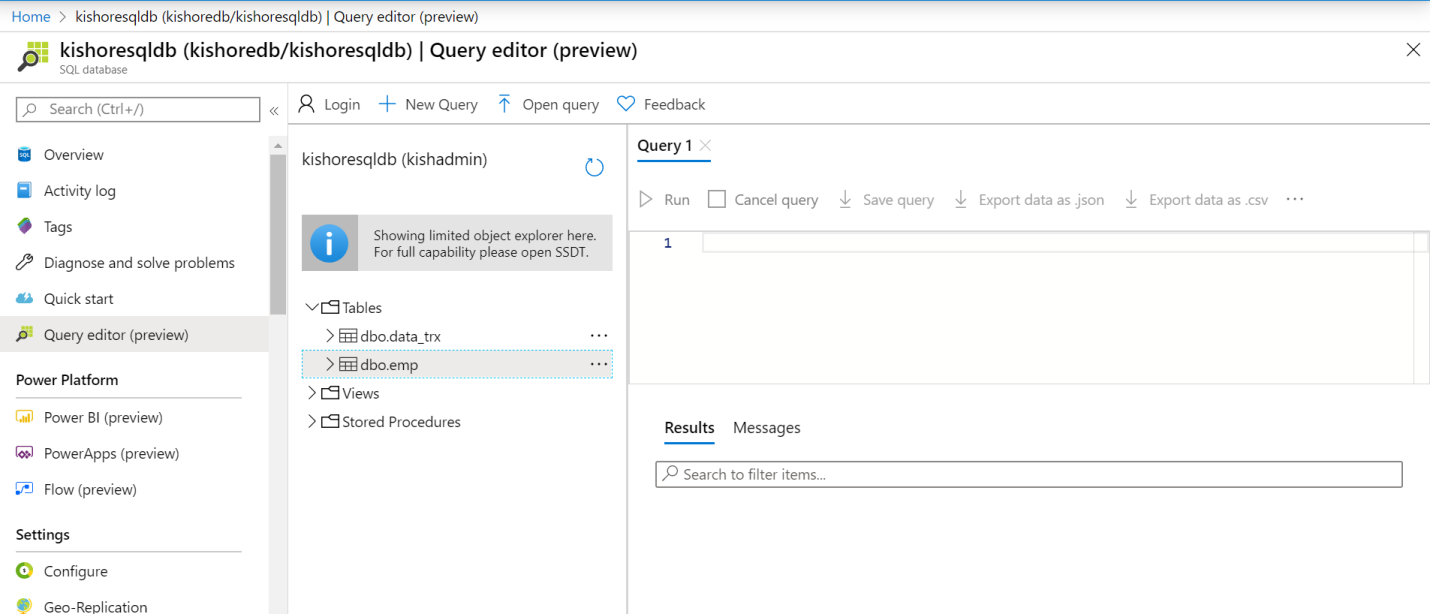
Navigate to the TimerTriger logs console on azure portal



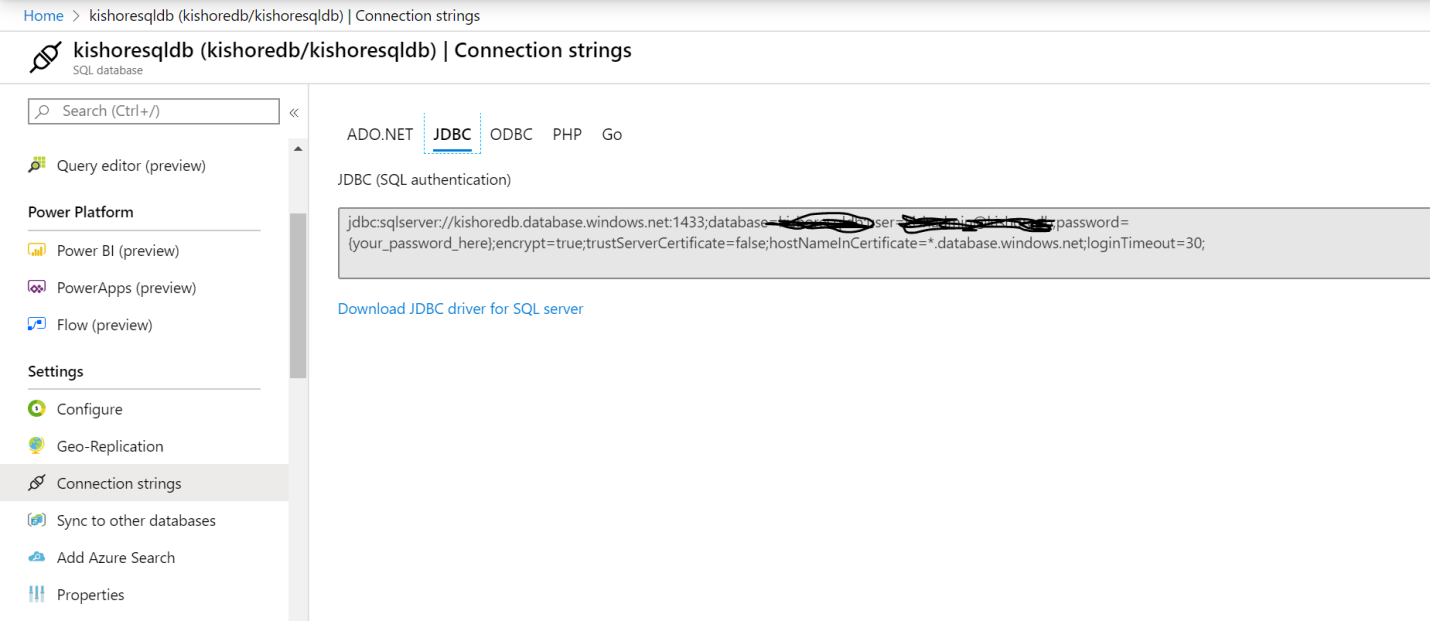


**HttpTrigger Function integrate with Azure SQL**

## 32. Create the SQL Server & SQL DB in Azure. Add a new data\_trx table to kishoresqldb

****

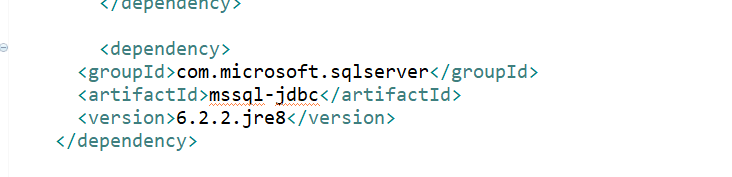
**Copy the connection String value from DB**

****

**Add the connection String value as URL.**

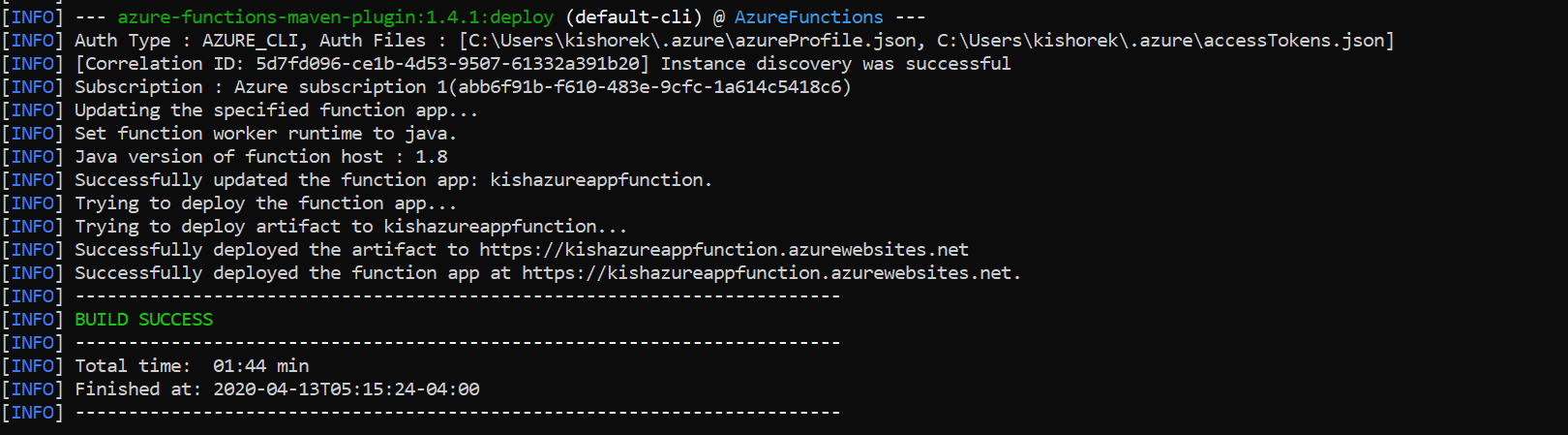
****

**Add the following dependency to the POM.xml**

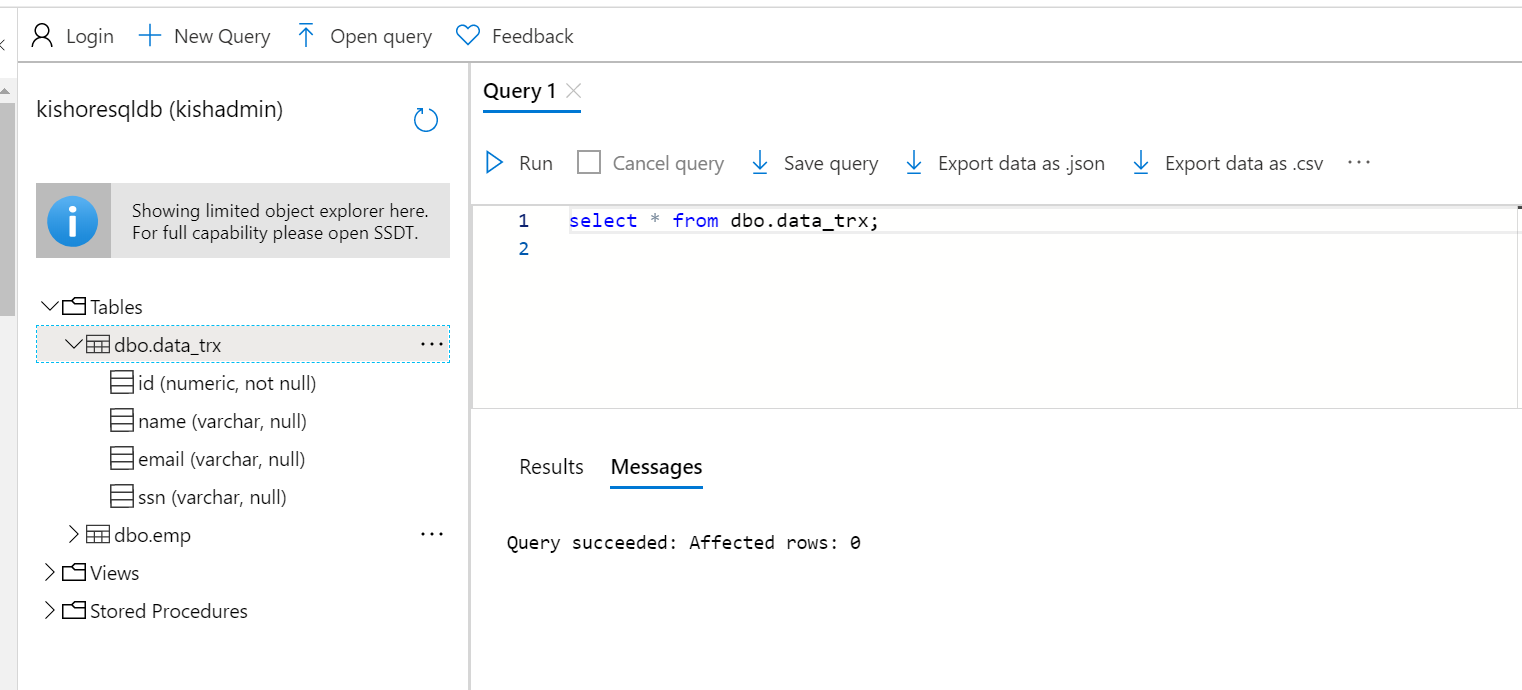
****

Deploy the Azure function use the following command

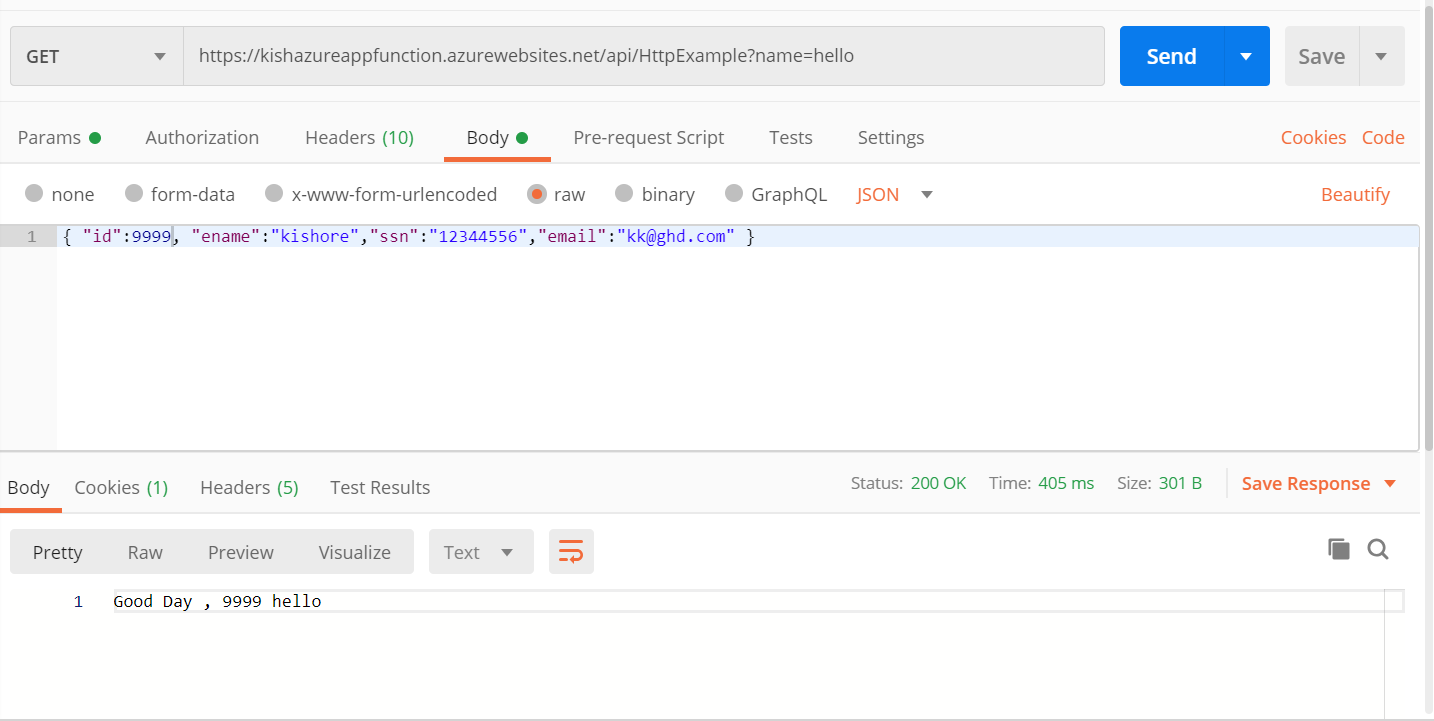
mvn install package azure-functions:deploy



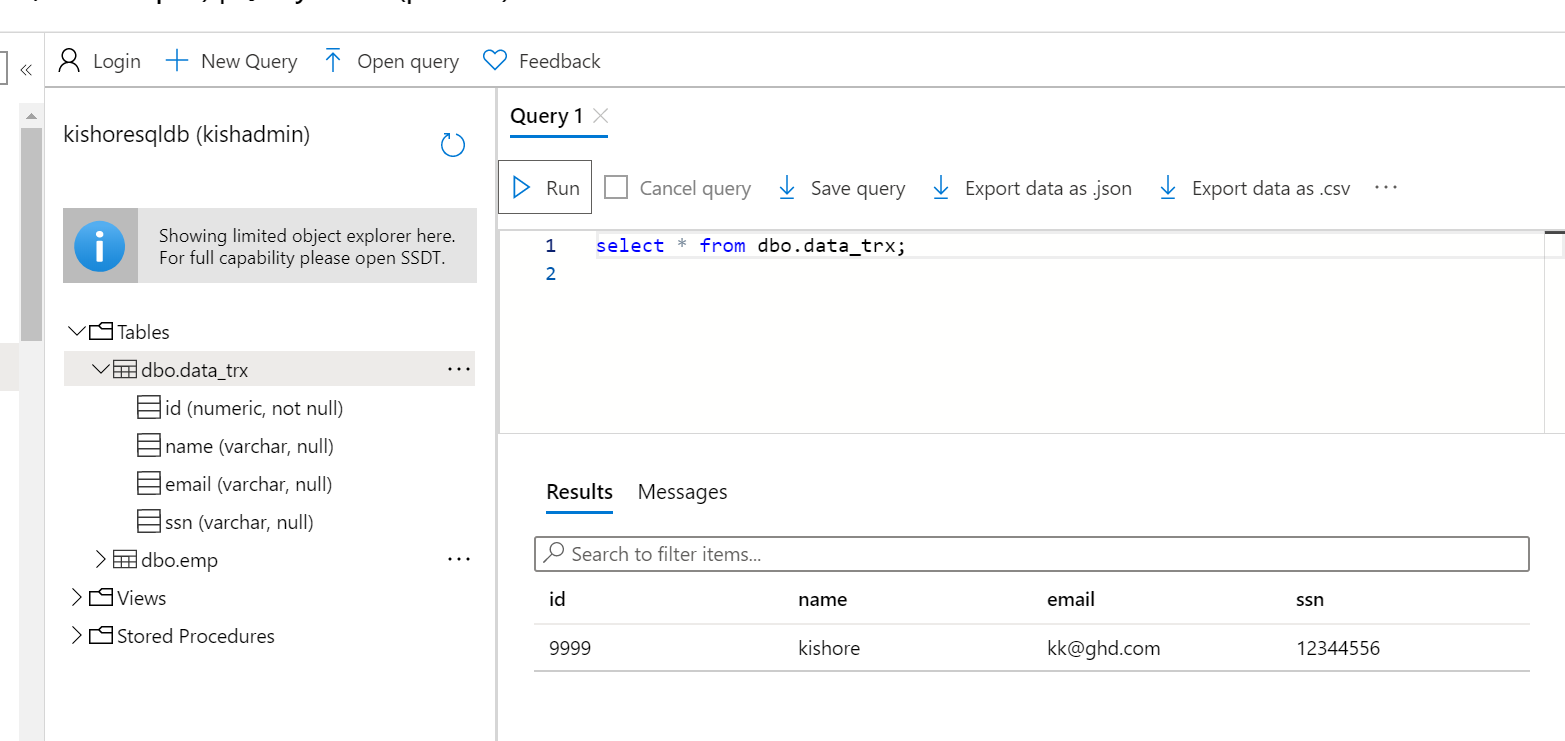
**Select query from data\_trx table before request**

****

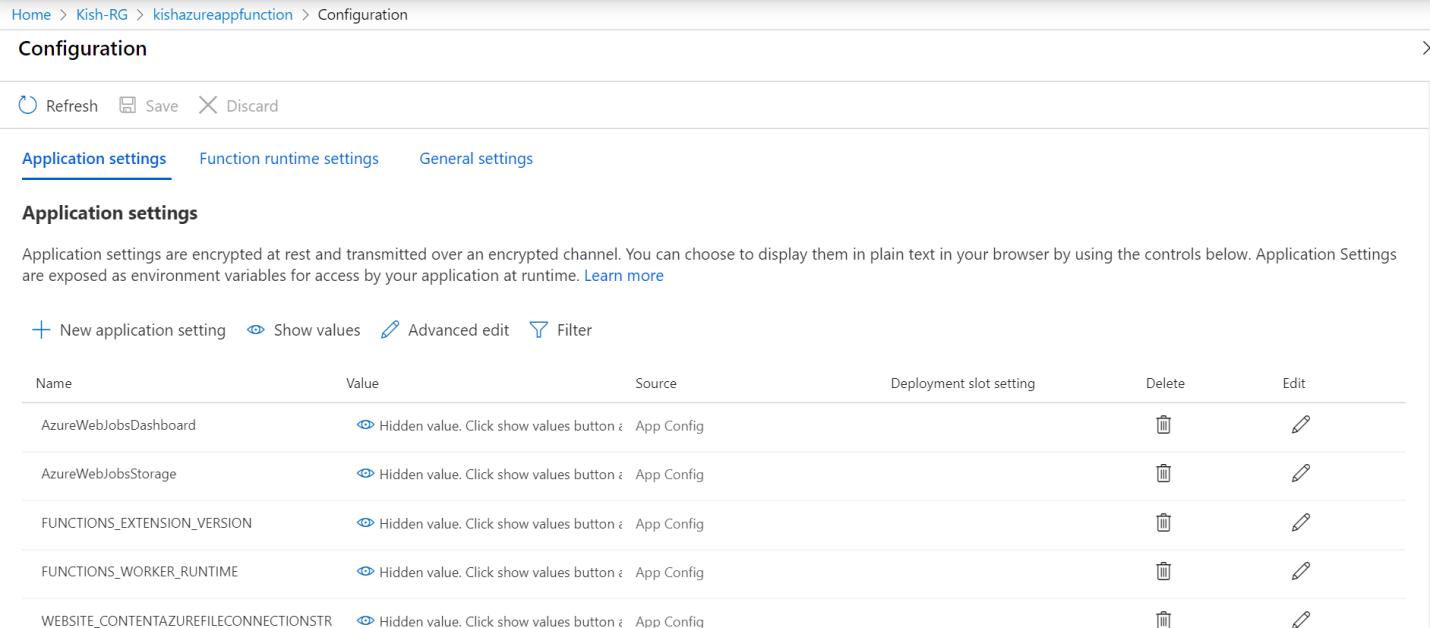
Send a new request to Http Trigger

****

**Select query from data\_trx table After request**

****

**\*\* We can use the existing store account instead of new one by updating the AzureWebJobsStorage value.**

****