

Azure Service Bus, Logic App & SQL DB

Hackathon Instructions Guide

June 2016

© 2016 Microsoft Corporation. All rights reserved. This document is confidential and proprietary to Microsoft. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

Contents

[Objective 3](#_Toc454636458)

[Prerequisites 3](#_Toc454636459)

[Overview 3](#_Toc454636460)

[Steps 3](#_Toc454636461)

[Step 1 Create a Service Bus Namespace 3](#_Toc454636462)

[Step 2 Install the MessagePublisher client 4](#_Toc454636463)

[Step 3 Run MessagePublisher client 5](#_Toc454636464)

[Create Resources in the Azure Portal 6](#_Toc454636465)

[Step 4 Create a Resource Group 6](#_Toc454636466)

[Step 4a Create a Storage Account 8](#_Toc454636467)

[Step 5 Create a Logic App 8](#_Toc454636468)

[Step 6 Design Logic App 10](#_Toc454636469)

# Objective

This lab will show you how to setup Azure Service Bus, Azure Logic App and Azure SQL DB. After completing this lab you should get an idea how easy it is to setup and develop Azure solutions.

Duration: 1.5 to 2 hours

# Prerequisites

You will need an Azure Subscription. Its recommended to use MSDN or Personal account for this hackathon.

You will need SSMS or Microsoft Visual Studio 2015

# Overview

Login to azure portal and setup a Azure Service Bus, Logic App and Azure SQL DB. Configure Azure Service Bus to send records to Logic App. Configure and develop a workflow in Logic App to gran this queue and insert them into a table. Configure Azure SQL DB, firewall and create a single table that will store the data.

Optional: OMS

# Steps

## Step 1 Create a Service Bus Namespace

At the time of writing, Azure Service Bus is only supported in the classic portal.

1. Login to the Azure classic portal -> https://manage.windowsazure.com/
2. Navigate to SERVICE BUS -> click CREATE
3. In the Add a new namespace modal dialog:
   1. Provide a unique NAMESPACE NAME; make note of this name as it will be used in subsequent steps
   2. Under TYPE, verify that MESSAGING is selected
   3. Under MESSAGING TIER, verify that STANDARD is selected
   4. Under REGION, specify West US
   5. Click on the check icon to create the Service Bus namespace
4. Click on the newly created namespace -> click CONFIGURE
   1. Copy and set aside the POLICY NAME; it will be RootManageSharedAccessKey by default
   2. Copy the set aside the PRIMARY KEY value
   3. Values 4a and 4b will be used to create a connection string in a subsequent step

## Step 2 Install the MessagePublisher client

This client will be used to publish messages to your newly created Azure Service Bus namespace. The client will also create a Service Bus topic and a Service Bus topic subscription. A sample transaction is comprised of a booking response envelope which contains a book request. Below is an example of a single JSON formatted transaction:

|  |
| --- |
| {  "Request": {  "BookingId": "10ff9b5d-747b-49b6-a509-86c6137e9108",  "Traveler": "James Jones",  "TransactionDate": "2016-06-25T18:35:16.3369898Z"  },  "BookingId": "10ff9b5d-747b-49b6-a509-86c6137e9108",  "Traveler": "James Jones",  "TransactionDate": "2016-06-25T18:35:16.3369898Z"  } |

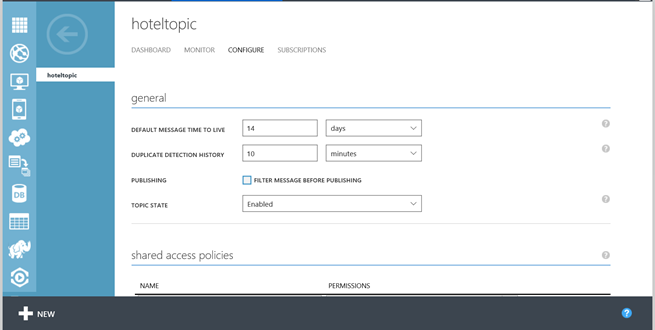
1. Download and install the MessagePublisher client
   1. Navigate to <https://github.com/tempehipopod/hackathon-logicapp> -> click Clone or download button -> choose Download ZIP
   2. Execute MessagePublisher.msi; make note of the installation directory
2. Configure MessagePublisher client by navigating to the installation directory -> with your favorite text editor (try Notepad++ or SublimeText), edit the App.config file. Locate the appSettings at the bottom of the file. The appSettings will appear as shown:

|  |
| --- |
| <appSettings>  <add key="topicname" value="**hoteltopic**"/>  <add key="subscriptionname" value="**transactions**" />  <!-- Service Bus specific app setings for messaging connections -->  <add key="Microsoft.ServiceBus.ConnectionString"  value="Endpoint=sb://**[your namespace]**.servicebus.windows.net;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=**[your secret]**"/>  </appSettings> |

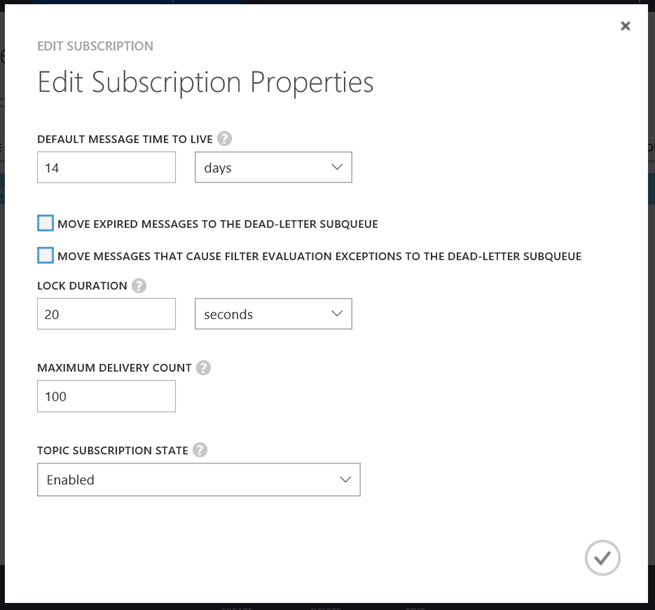
1. You can choose to leave topicname and subscriptionname values as is, or, you can change the value attribute (i.e. “hoteltopic” and/or “transactions”)
2. Replace [your namespace] with the Service Bus namespace created in Step 1-3a
3. Replace [your secret] with the value copied in Step 1-4b

## Step 3 Run MessagePublisher client

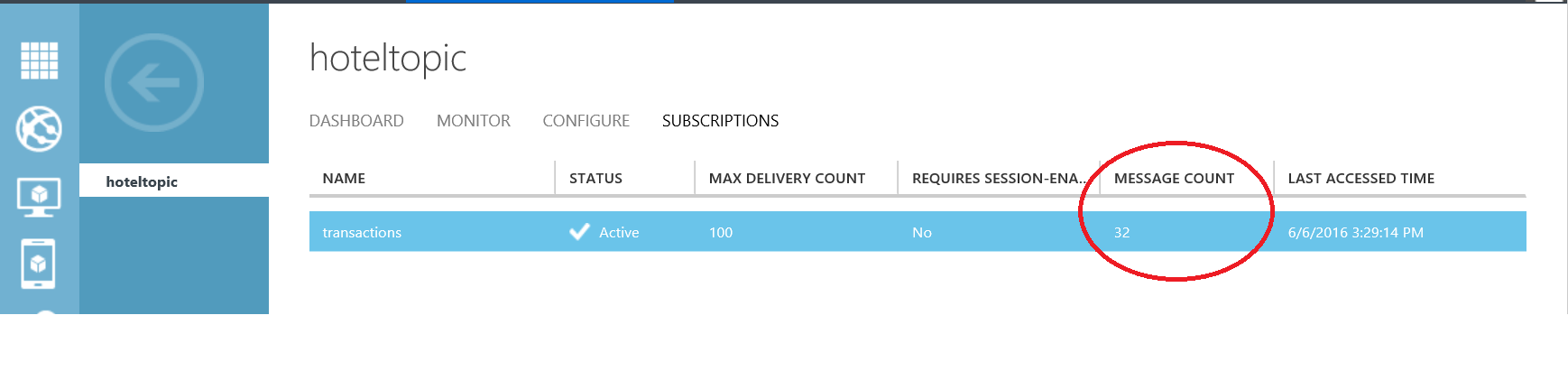
1. Run the message client by double-clicking MessagePublisher.exe
2. Verify that the client created a Service Bus topic and subscription
   1. Navigate to the SERVICE BUS page of the Azure classic portal (<https://manage.windowsazure.com/microsoft.onmicrosoft.com#Workspaces/ServiceBusExtension/namespaces>)
   2. Click on the namespace created in Step 1 -> click TOPICS -> Verify that a topic named hoteltopic (or the topic name you specified in step 2-2a) has been created
      1. Click on the topic -> click CONFIGURE; make note of the available settings as shown below



* + 1. Click SUBSCRIPTIONS -> Verify that a subscription named transactions (or the subscription name you specified in step 2-2a) has been created
    2. Click the edit icon at the bottom of the page. Make note of the available settings as shown below



1. In the MessagePublisher client, type 1 in the console and hit enter
   1. Within the Azure classic portal, navigate to the Service Bus topic subscription and verify that a message has been created; see below

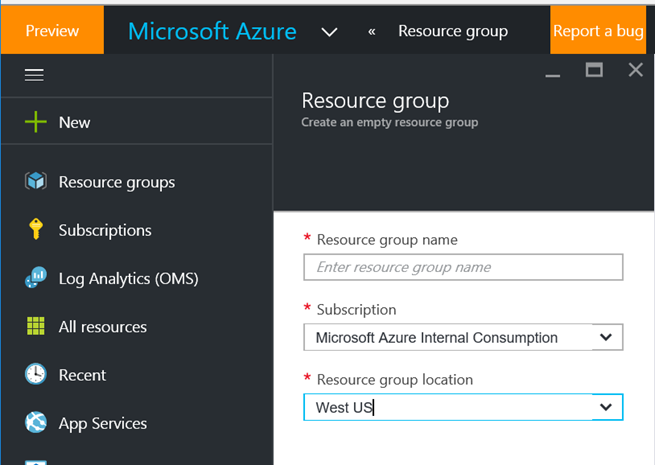


## Create Resources in the Azure Portal

Resource groups provide a container for a functional grouping of one or more Azure services. Resource groups make it convenient to manage (view, edit, delete, update, deploy) individual resources or all resources within a group. Resource groups are a key element of the Azure Resource Manager (ARM) model. This is also known as Azure V2. With the exception of Azure Service Bus, all services within this lab are managed using ARM/V2.

### Step 4 Create a Resource Group

1. Login to the Microsoft Azure Portal (<https://portal.azure.com>)
2. Click Resource Groups in the left navigation pane
3. In the Resource Groups blade, click + to create a new resource group
4. In the Resource group blade, provide the following information:
   1. Resource group name (consider adding an -rg suffix)
   2. Subscription – choose a subscription if you have multiple subscriptions
   3. Resource group location – select the location nearest you (ex. West US)



1. Click Create (allow a few seconds for the resource group to be created)

### Step 4a Create a Storage Account

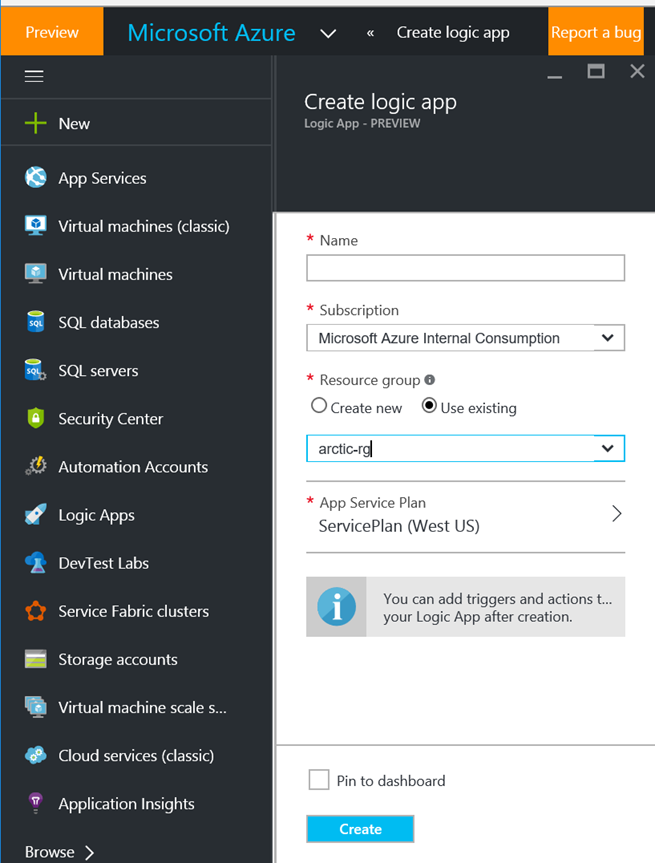
This storage account will be used to store all diagnostics information.

1. Within the Azure Portal, navigate to Storage accounts
2. Within the storage accounts blade, click add
3. In the Create storage account blade, provide the following:
   1. Name – must be unique, all lower case, no special characters, 3-24 characters long
   2. Replication – Locally-redundant storage (LRS)
   3. Resource group – select “Use existing” and specify the resource group created in Step M-N
   4. Location – specify a location near you (ex. West US).

This same location must be used when you create a Logic App below.

### Step 5 Create a Logic App

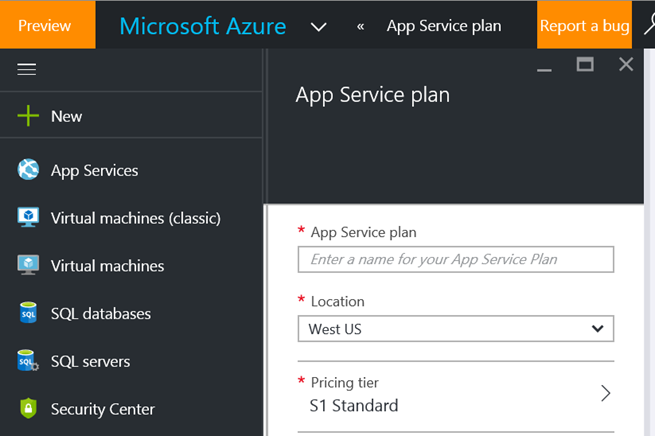
1. Login to the Microsoft Azure Portal (<https://portal.azure.com>)
2. Click Browse -> type “Logic App” in the Filter field
3. Click the star icon to add it as a favorite -> Select Logic Apps
4. In the Logic Apps blade, provide the following information:



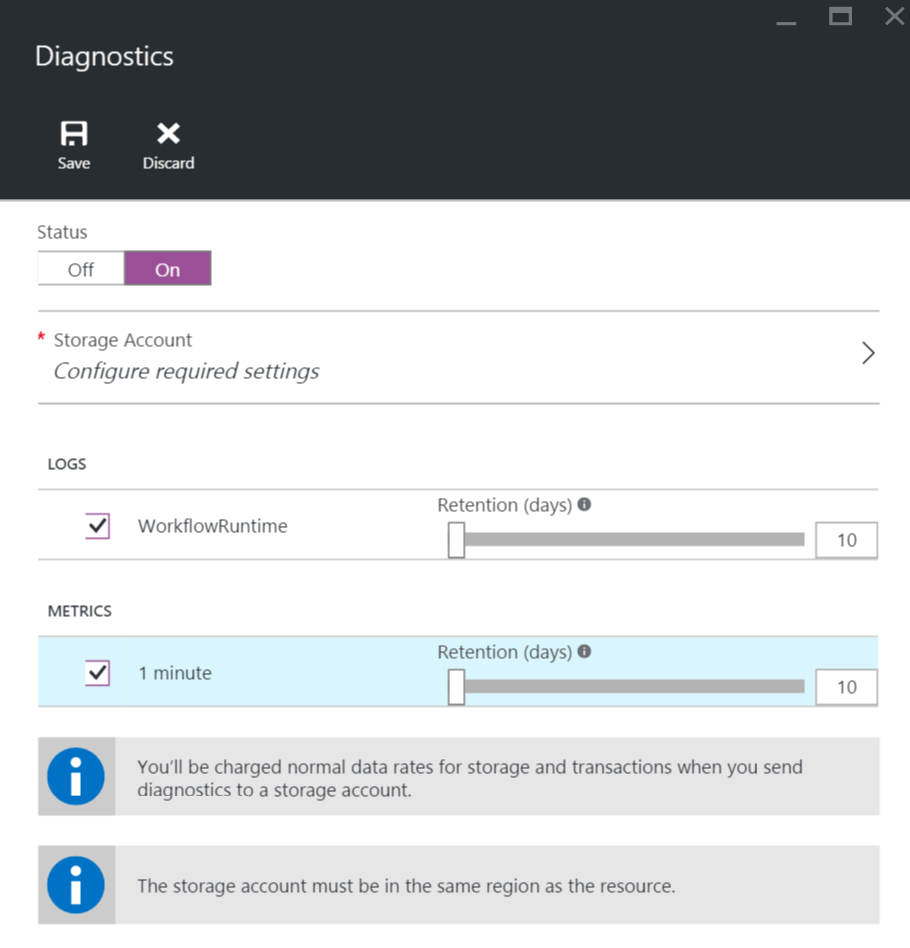
* 1. Name
  2. Subscription
  3. Resource group
     1. Click Use existing
     2. Select the resource group created in Step 4
  4. App Service Plan

The App Service Plan must be a Standard plan. You may use an existing Standard plan, or create a new plan. If you choose to create a new plan, provide the following information in the App Service plan blade:

* + 1. App Service plan – consider adding a -asp suffix
    2. Location – specify a location near you (ex. West US)
    3. Pricing tier – this must be at least **S1 Standard**. Standard pricing tier and above will allow you to specifier more granular polling frequencies for Logic App triggers.



1. Click Create. Allow for a few seconds for provisioning to complete.
2. Enable Diagnostics
   1. In Logic App blade, click on the “Runs success latency in the past 3 hours” display; doing so will bring up the Diagnostics blade.
   2. Within the Diagnostics blade:
      1. Set Status to On
      2. Storage Account – choose the storage account that was previously created. Note that the location must match the location of the Logic App Service plan.
      3. Ensure that both checkboxes are checked for Logs and Metrics; specify a retention of 10 days for both.

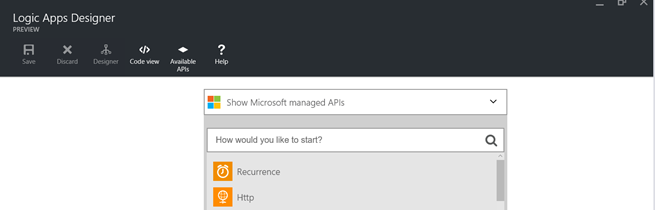


# Design a Logic App to receive mesages

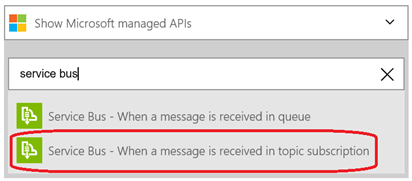
## Step N Configure the Azure Service Bus Connector

The Azure Service Bus Connector will be used to periodically retrieve messages from a designated topic subscription.

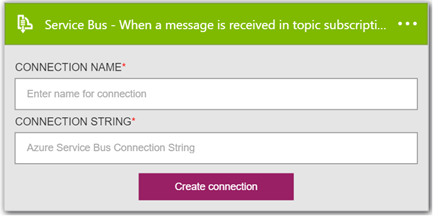
1. Login to Microsoft Azure Portal -> navigate to Logic Apps -> Select the Logic App created in Step 5. This will open the Logic App editor.



1. Enter “Service Bus” in place of “How would you like to start?” -> Select the second option shown below



Once selected, you will be asked to create a connection as shown:



* 1. Provide a CONNECTION NAME
  2. CONNECTION STRING – copy the value of the Microsoft.ServiceBus.ConnectionString appSetting value configured in Step 2-2. Example:

|  |
| --- |
| Endpoint=sb://**[your namespace]**.servicebus.windows.net;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=**[your secret]** |

* 1. Click “Create connection”

1. Back in the Service Bus topic subscription dialog, provide the following:
   1. TOPIC NAME – use the same topic name specified in Step 2-2
   2. TOPIC SUBSCRIPTION NAME – use the same topic subscription name specified in Step 2-2
2. Add an additional action so we can save the Logic App.
3. Click the + button -> click “Add an action”
4. Select Http. In the dialog, specify the following:
   1. METHOD – select GET
   2. URI – select any URI
5. Click the save button in the Logic Apps Designer.

## Step N Adjust the Service Bus Trigger Polling Frequency

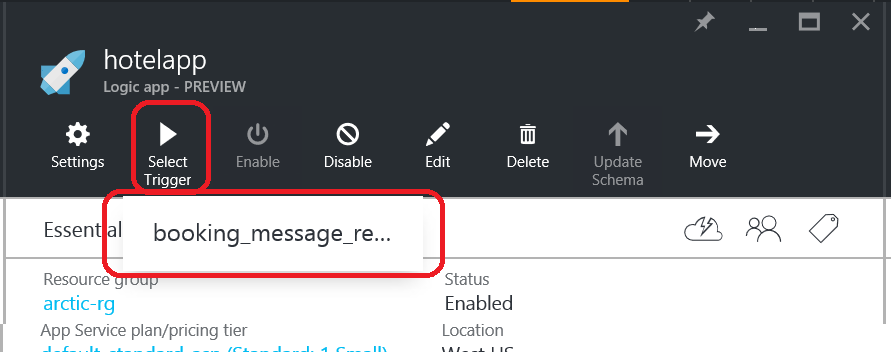
1. Switch to Code view </>
2. Change the trigger’s name to “booking\_message\_trigger”. Look for the following section and replace the bold text.

|  |
| --- |
| "triggers": {  "**When\_a\_message\_is\_received\_in\_topic\_subscription**": { |

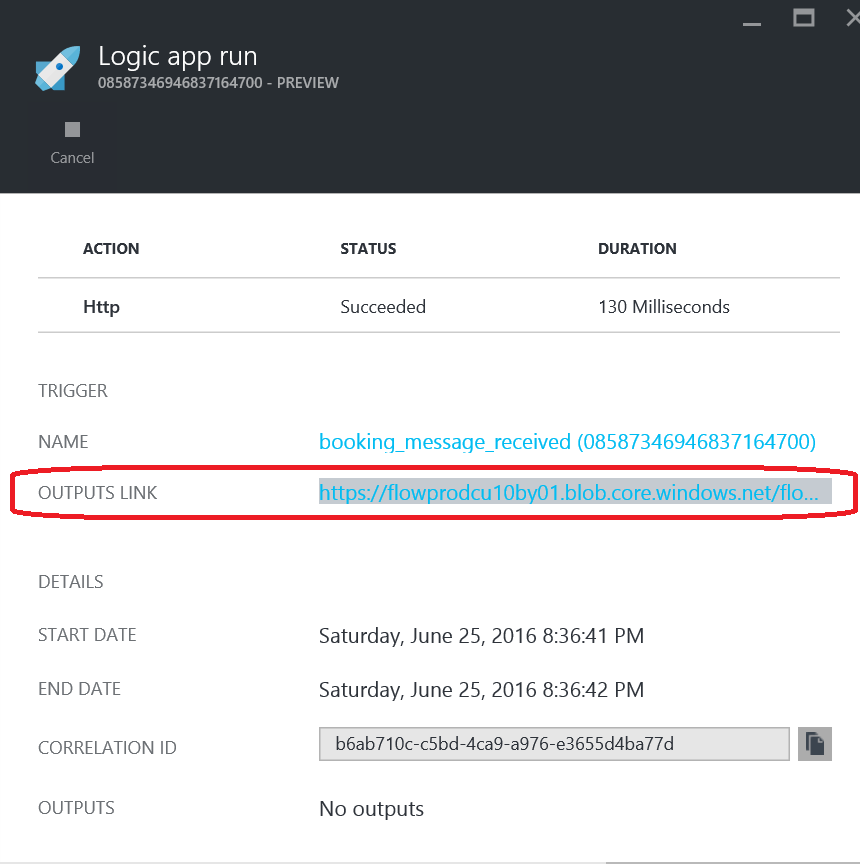
1. Change the trigger’s interval from every hour to every 15 seconds by changing the “recurrence” element as shown:

|  |
| --- |
| "recurrence": {  "frequency": "**Second**",  "interval": 15  }, |

1. Click save. Close the Logic App Designer blade.
2. In the main Logic App blade, click on the “booking\_message\_trigger” within the All triggers box. Notice the entries within the Trigger histories table. Close the “booking\_message\_trigger” blade.
3. Using the MessagePublisher client, send messages to the service bus.
4. Fire the trigger manually by clicking the Select Trigger icon -> click booking\_message\_received



1. Click on the All runs box to view the All runs pane -> click on a single run to view the Logic app run blade -> click on the OUTPUS LINK to view the JSON message



Below is a sample JSON message submitted by the MessagePublisher client.

|  |
| --- |
| {  "statusCode": 200,  "headers": {  "Pragma": "no-cache",  "Retry-After": "0",  "x-ms-request-id": "b6ab710c-c5bd-4ca9-a976-e3655d4ba77d",  "Cache-Control": "no-cache",  "Date": "Sun, 26 Jun 2016 03:36:40 GMT",  "Location": "https://logic-apis-westus.azure-apim.net/apim/servicebus/00707d7fafd14a28b4104c776888a588/hotel/subscriptions/booking/messages/head?triggerstate=f0662092-bb1c-4549-87ec-76b306c4141e",  "Set-Cookie": "ARRAffinity=a0d9ab6dc8facbfb6adc55a4577a3242faedf7e86b07d9afecd63b6b5836935b;Path=/;Domain=127.0.0.1",  "Server": "Microsoft-IIS/8.0,Microsoft-HTTPAPI/2.0",  "X-AspNet-Version": "4.0.30319",  "X-Powered-By": "ASP.NET",  "Content-Length": "1146",  "Content-Type": "application/json",  "Expires": "-1"  },  "body": {  "**ContentData**": "eyJSZXF1ZXN0Ijp7IkJvb2tpbmdJZCI6ImE3NDhjM2QyLWZlOGEtNGE3Yi05N2RhLTYxMTczYTdiM2EyZSIsIlRyYXZlbGVyIjoiR2VvcmdlIEplZmZlcnNvbiIsIlRyYW5zYWN0aW9uRGF0ZSI6IjIwMTYtMDYtMjZUMDM6MzY6MzUuNTU1OTI4WiJ9LCJCb29raW5nSWQiOiJhNzQ4YzNkMi1mZThhLTRhN2ItOTdkYS02MTE3M2E3YjNhMmUiLCJUcmF2ZWxlciI6Ikdlb3JnZSBKZWZmZXJzb24iLCJUcmFuc2FjdGlvbkRhdGUiOiIyMDE2LTA2LTI2VDAzOjM2OjM1LjU1NTkyOFoifQ==",  "ContentType": "text/plain",  "ContentTransferEncoding": "Base64",  "Properties": {  "CorrelationId": "bookingresponse",  "DeliveryCount": "1",  "EnqueuedSequenceNumber": "6",  "EnqueuedTimeUtc": "2016-06-26T03:36:35Z",  "ExpiresAtUtc": "9999-12-31T23:59:59Z",  "LockedUntilUtc": "2016-06-26T03:37:41Z",  "LockToken": "f0662092-bb1c-4549-87ec-76b306c4141e",  "MessageId": "a748c3d2-fe8a-4a7b-97da-61173a7b3a2e",  "ScheduledEnqueueTimeUtc": "0001-01-01T00:00:00Z",  "SequenceNumber": "3",  "Size": "434",  "State": "Active",  "TimeToLive": "9223372036854775807"  },  "MessageId": "a748c3d2-fe8a-4a7b-97da-61173a7b3a2e",  "To": null,  "ReplyTo": null,  "ReplyToSessionId": null,  "Label": null,  "ScheduledEnqueueTimeUtc": "0001-01-01T00:00:00Z",  "SessionId": null,  "CorrelationId": "bookingresponse",  "TimeToLive": "9223372036854775807"  }  } |

Notice that the ContentData element is Base64 encoded.

# Create an Azure SQL Database staging table