Azure Data Processing

LAB Overview

This lab introduces you how to build data processing using Azure services. You will use Azure Function as a metrics generator. Event hub will be main message aggregator, Stream Analytics will transmit data to Service Bus which triggers the another Function and send data to Azure Redis Cache.

Task 1: Create an Event Hub namespace

In this section, you will learn how to create an Event Hub namespace from Azure Portal.

1. Sign in the Azure portal at
   https://portal.azure.com
2. On the left **Hub** menu click on **Create a resource.**
3. On the **New** blade click on **Internet of Things** and select **Event Hubs.**
4. On the **Create namespace** blade provide the following configurations:

   - **Name:** event-hub-student0X
   - **Pricing Tier:** Standard
   - **Subscription:** XXXXXX
   - **Resource group:** student0X
   - **Location:** West Europe
   - **Throughput Unit:** 1
   - **Select** Enable auto-inflate
   - **Specify Upper Limit:** 20

When, you finish click on button **Create.**
5. Next step the Azure Event Hub namespace will be deployed.

Task 2: Add Event Hubs to namespace

In this section, you will learn how to add a new instance to namespace of created Azure Event Hub from previous task.

1. After successful deployment of Azure Event Hub, go to created service **event-hub-student0X**.
   You can search the service using field search **Search resources, services and docs** which is located at the top of Azure Portal.
2. On the **Event Hubs Namespace** page click on the **Event Hub** button marked with a plus sign.
3. On the **Create Event Hub** blade provide the following configurations:

   - **Name**: dataBroker
   - **Partition Count**: 2
   - **Message Retention**:
   - **Capture**: Off

When you finish, click on button **Create.**

4. Next step the Event Hub will be added to namespace.
5. Add two more event hubs in the same way: **metrics** and **events.**

Task 3: Add consumer groups to Event Hub

In this section, you will learn how to add consumer group to instance of Event Hub from Azure Portal.

1. On the main **Overview** page of **Event Hubs Namespace**, click on **Event Hubs** from the left menu.
2. On the **Event Hubs** page click on created instance of Event Hub - **dataBroker**.
3. On the instance of Event Hub – **dataBroker** page click on **Consumer groups** from the left menu.
4. On the **Consumer Group** page click on button **Consumer Group** marked with plus assign and add two groups named:

   - stream-analytics

Task 4: Create Azure Functions

In this section, you will learn how to create Azure Functions from Azure Portal.

If you already have Azure Functions resource, then skip this task.

1. On the left **Hub** menu click **App Services.**
2. On the **App Services** blade click on the **Add** button marked with a plus sign.
3. Type **Function app** in search filed.
4. After searching, click on position **Function app** and click on **Create.**
5. On the **New** blade of Function app provide the following configuration:

    - **App name:** function-app-student0X
    - **Subscription:** XXXXXXXX
    - **Resource Group:**
        - **Use existing:** student0X
    - **Os:** Windows
    - **Hosting Plan:** Consumption Plan
    - **Location:** West Europe
    - **Storage:** Create new
    - **Application Insight:** Off

When you finish, click on button **Create.**

6. Next step the Azure Functions will be deployed.

Task 5: Create an instance of function

In this section, you will learn how to create an instance of function triggered by timer and how to add custom definition of code.

1. After successful deployment of Azure Functions, on the left **Hub** menu click **Function apps.**
2. Click on your resource of Azure Functions - **function-app-student0X.**
3. Familiarize yourself with the main dashboard of Azure Functions**.**
4. On the main **Overview** page click on **Functions** from the left side menu.
5. On the **Functions** page click on the **New function** button marked with a plus sign.
6. Click on **Time trigger** method.
7. On the **Time Trigger New Function** blade provide the following configurations:

    - **Language**: C#
    - **Name**: MessageGenearator
    - **Schedule**: */5 * * * * *

And click on **Create.**

8. On the **MessageGenearator** page function change code in **run.csx** file:

```
#r "Newtonsoft.Json"
using System;
using Newtonsoft.Json;
public static string Run(TimerInfo myTimer, TraceWriter log)
{
  log.Info($"C# Timer trigger function executed at: {DateTime.Now}");
  Random value = new Random();
  string json = JsonConvert.SerializeObject( new { deviceId= value.Next(0,10), a= value.Next(0,1000), b=
value.Next(0,100), Time = DateTime.Now});
  log.Info($"JSON output: {json}");
  return json;
}
```

And click on **Save.**

Task 6: Add Event Hub output to Function

In this section, you will learn how to add output from Event Hub to Azure Functions.

1. On the **MessageGenearator** page function click on **Integrate.**
2. On the **Integrate** page click on **New Output** and select **Azure Event Hubs.**
3. Next on **Azure Event Hubs Output** section provide the following configurations:

   - **Event parameter name:** $return
   - **Select** Use Function return value
   - **Event Hub name:** dataBroker
   - **Event Hub connection:**
     - **Select** Event Hub
     - **Namespace:** event-hub-student0X
     - **Event Hub:** databroker
     - **Policy:** RootMangeSharedAccessKey
       And click on **Select.**

And click on **Save.**

Task 7: Create Azure Service Bus

1. On the left **Hub** menu click on **Create a resource.**
2. On the **New** blade click on **Internet of Things** and select **Service Bus.**
3. On the **Create namespace** blade provide the following configurations**:**

    - **Name**: sb-student0X
    - **Pricing Tier**: Standard
    - **Resource Group**:
        - **Use existing**: student0X
    - **Location:** West Europe

When you finish click on **Create.**

Task 8: Add queue to Azure Service Bus

1. On the main **Overview** page of created Azure Service Bus click on **Queues** from the left menu.
2. Next click on **Queue** button marked with plus assign.
3. On the **Create queue** blade provide the following configurations:

    - **Name:** metrics
    - **Max size:** 1 GB
    - **Message time to live (default):** 14 days
    - **Lock duration:** 30 seconds

      And select **Enable partitioning.**

When you finish click on **Create.**

Task 8: Create Stream Analytics

In this section you will learn how to create Azure Stream Analytics.

4. On the left **Hub** menu click on **Create a resource.**
5. On the **New** blade click on **Data + Analytics** and select **Stream Analytics job.**
6. On the **New Stream Analytics job** blade provide the following configurations**:**

   - **Job name**: sa-student0X
   - **Subscription: XXXXX**
   - **Resource Group:** student0X
   - **Location:** West Europe
   - **Hosting Environment:** Cloud

When, you finish click on button **Create.**

7. Next step the Azure Stream Analytics will be deployed.

Task 9: Add Event Hub input to Stream Analytics

In this section you will learn how to add input source of Event Hub in Stream Analytics.

1. On the main **Overview** page of **Stream Analytics**, click on **Inputs**.
2. On the **Inputs** page click on button **Add Stream inputs** and select **Event Hub**.
3. On the **Event Hub new input** provide the following configurations:

   - **Input alias:** EventHub
   - **Select Event Hub from your subscription**
   - **Event Hub namespace:** event-hub-student0X
   - **Event Hub name**
     - **Use existing:** databroker
   - **Event Hub consumer group:** stream-analytics
   - **Event serialization format:** JSON
   - **Encoding: UTF-8**
   - **Event compression type:** None

And click on **OK.**

4. Next go on created Event Hub namespace **event-hub-student0X** from previous tasks and click on **Shared Access policies.**
5. On the **Shared Access policies** page click on **RootManageSharedAccessKey.**
6. On the **Shared Access policies** blade copy value from **Primary key** field.
7. Go on main main **Overview** page of **Stream Analytics**, click on **Inputs.**
8. On **Inputs** page click on created **EventHub** input and paste value of **Primary key** from step 6 and click on **Save.**
9. On **EventHub Input Details** click on **Test** to check the connection with EventHub.

Task 10: Add Service Bus output to Stream Analytics

In this section, you will learn how to add output source of events and metrics Event Hub in Stream Analytics.

1. On the main **Overview** page of **Stream Analytics**, click on **Outputs**.
2. On the **Outputs** page click on button **Add** and select **Service Bus Queue**.
3. On the **Service Bus queue New output** provide the following configurations:

    - **Output alias:** Metrics
    - **Select queueHub from your subscription**
    - **Service Hub namespace:** sb-student0X
    - **Queue name**
        - **Use existing:** metrics
    - **Queue policy name:** RootManageSharedAccessKey
    - **Event serialization format:** JSON
    - **Encoding:** UTF-8
    - **Format:** Line separeted

And click on **Save.**

4. On **ServiceBus Output Details** click on **Test** to check the connection with ServiceBus.

Task 11: Add query to Stream Analytics

In this section you will learn how to add SQL query in Azure Stream Analytics which processing all data from Event Hub to Service Bus Queues.

1. On the main **Overview** page of **Stream Analytics** click on **Stop** to stopping Stream Analytics job.
2. On the main **Overview** page of **Stream Analytics**, add below lines of SQL code clicking on **Edit Query** in **Query** section:

```
SELECT a, b, CAST(Time as datetime) as Time
INTO Metrics
FROM EventHub
```

And click on **Save.**

3. On the main **Overview** page of **Stream Analytics** click on **Start** to run Stream Analytics job.

Task 12: Create Azure Redis Cache

In this section, you will learn how to create Azure Redis Cache.

1. On the left **Hub** menu click on **Create a resource.**
2. On the **New** blade click on **Databases** and select **Redis Cache.**
3. On the **New Redis Cache** blade provide the following configurations:

    - **DNS Name:** redis-cache-student0X
    - **Subscription:** XXXXXX
    - **Resource Group:**
        - **Use existing:** student0X
    - **Location:** West Europe
    - **Pricing Tier:** Basic C0

When you finish click on **Create.**

Task 13: Create an instance of function

1. Click on your resource of Service Bus – **sb-student0X.**
2. On the main **Overview** page click on **Queues** from the left menu.
3. On the **Queues** page click on created **metrics** queue.
4. On the **metrics queue** blade click on **Shared access policies** and click on **Add** button marked with a plus sign and provide the following configurations:
    - **Policy name:** Listen

Select **Listen** and click on **Create.**

5. Then click on created policy and copy value from the field **Primary Connection String.**
6. Click on your resource of Azure Functions - **function-app-student0X.**
7. On the main **Overview** page click on **Functions** from the left side menu.
8. On the **Functions** page click on the **New function** button marked with a plus sign.
9. Click on **ServiceBusQueueTrigger** method.
10. On the **ServiceBusQueueTrigger** blade provide the following configurations:

    - **Language**: C#

- **Name**: ServiceBus
- **Queue**: metrics

And on section **Service Bus connection** click on **new** then on new window select **custom** and provide the following configurations:

- **Key**: ServiceBusConnectionString
- **Value**: Paste value from step 5 and modify enpoint without **EntityPath** substring, example:

  Modify endpoint from:

  Endpoint=sb://sb-student15.servicebus.windows.net/;SharedAccessKeyName=Listen;SharedAccesskey=dR7CIsR4aNEs1dXWCDQSCgJXmKsG0Nw5Q9Wlj26ikS4=;**EntityPath=metrics**

  To:

  Endpoint=sb://sb-student15.servicebus.windows.net/;SharedAccessKeyName=Listen;SharedAccesskey=dR7CIsR4aNEs1dXWCDQSCgJXmKsG0Nw5Q9Wlj26ikS4=

  And click on **Create.**

- **Access rights:** Listen
- **Queue name:** myQueue

When you finish click on **Create.**

11. On the **Overview** page of created Redis Cache – **redis-cache-student0X** click on **Access keys** and copy value from **Primary connection string.**
12. On the **Functions** page click on **Platform Features** and next click on **Application settings.**
13. On the **Application setting** click on **Add new setting** and add new field **RedisCacheEndpoint** and copy value from step 11, then click on **Save.**
14. On the **ServiceBusQueue** page function change code in **run.csx** file:

```
using System;
using System.Threading.Tasks;
using StackExchange.Redis;
using Newtonsoft.Json;
using System.Configuration;

public static void Run(string myQueueItem, TraceWriter log)
```

```
{
    log.Info($"C# ServiceBus queue trigger function processed message: {myQueueItem}");

    var msg = JsonConvert.DeserializeObject<MyOutput>(myQueueItem);

    if (msg == null)
    {
        log.Info("failed to convert msg to MyOutput");
        return;
    }

    IDatabase cache = Connection.GetDatabase();
    cache.StringSet(msg.deviceid, myQueueItem);
}

private static Lazy<ConnectionMultiplexer> lazyConnection = new Lazy<ConnectionMultiplexer>(() =>
{
    var endpoint = ConfigurationManager.AppSettings["RedisCacheEndpoint"];
    return ConnectionMultiplexer.Connect(endpoint);
});

public static ConnectionMultiplexer Connection
{
    get
    {
        return lazyConnection.Value;
    }
}

class MyOutput
{
    public string deviceid { get; set; }
    public string time { get; set; }
    public double a { get; set; }
    public double b { get; set; }
}
```

And click on **Save.**

Task 14: Redis Cache console.

1. On the **Overview** page of created Redis Cache – **redis-cache-student0X** click on **Console.**

2. On the **Console** type the command **Keys *** then you should see list of keys. To check one of them type command **Get “1”.**