Speech to Text

Step-By-Step Tutorial

IoT Cart

Contents

[Change Log 2](#_Toc36825460)

[Purpose 3](#_Toc36825461)

[Azure Services Used 3](#_Toc36825462)

[Required Hardware 3](#_Toc36825463)

[How to set up hardware 4](#_Toc36825464)

[Setting up the Cloud 4](#_Toc36825465)

[Getting Ready 5](#_Toc36825466)

[How to do it 5](#_Toc36825467)

[How it works 6](#_Toc36825468)

[Appendix 6](#_Toc36825469)

# Change Log

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Update** |
| 4/10/2020 | Connor Hagen | Created V1 |

# Purpose

This document will show you how to generate text from speech using a microphone, Raspberry Pi, and Azure resources such as Azure Speech service and SQL Server. The audio will be recorded from a microphone, translated to text in the cloud, then the text will be stored in a SQL database.

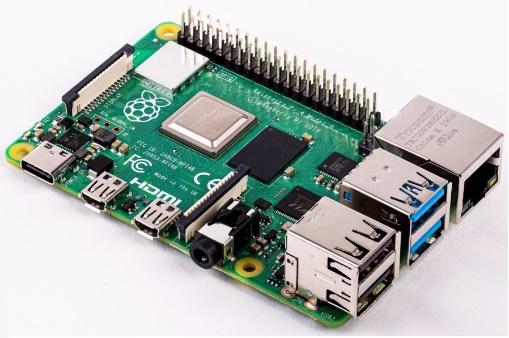
# Azure Services Used

1. Azure Speech service
2. SQL Server

# Required Hardware

You will need:

1. Raspberry Pi 2 or 3



1. USB Microphone

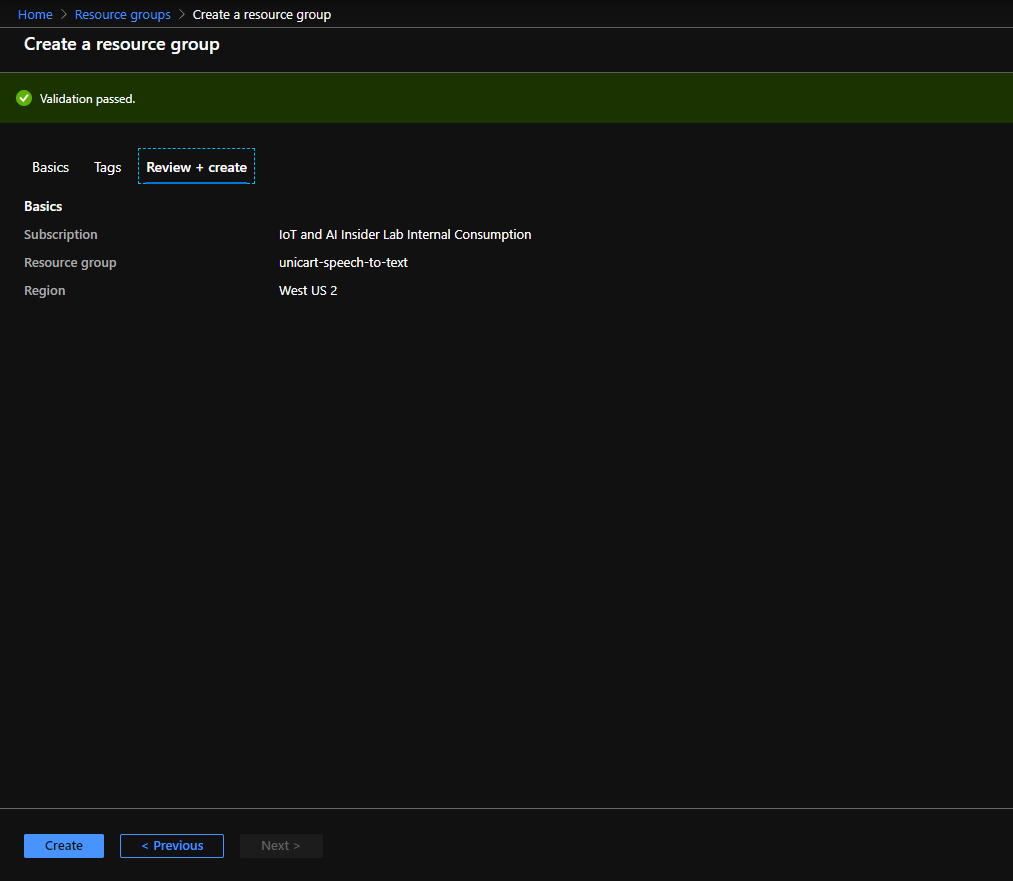
# How to set up the hardware

To make this you will need to connect the temperature sensor to the breadboard.

1. Connect USB Microphone to Raspberry Pi
2. Connect Raspberry Pi to power source

# Setting up the Cloud

**Step 1:** Create resource group in your subscription. This resource group will house the SQL Server and the Cognitive Services Speech resource.



**Step 2**: Create Cognitive Services Speech resource:

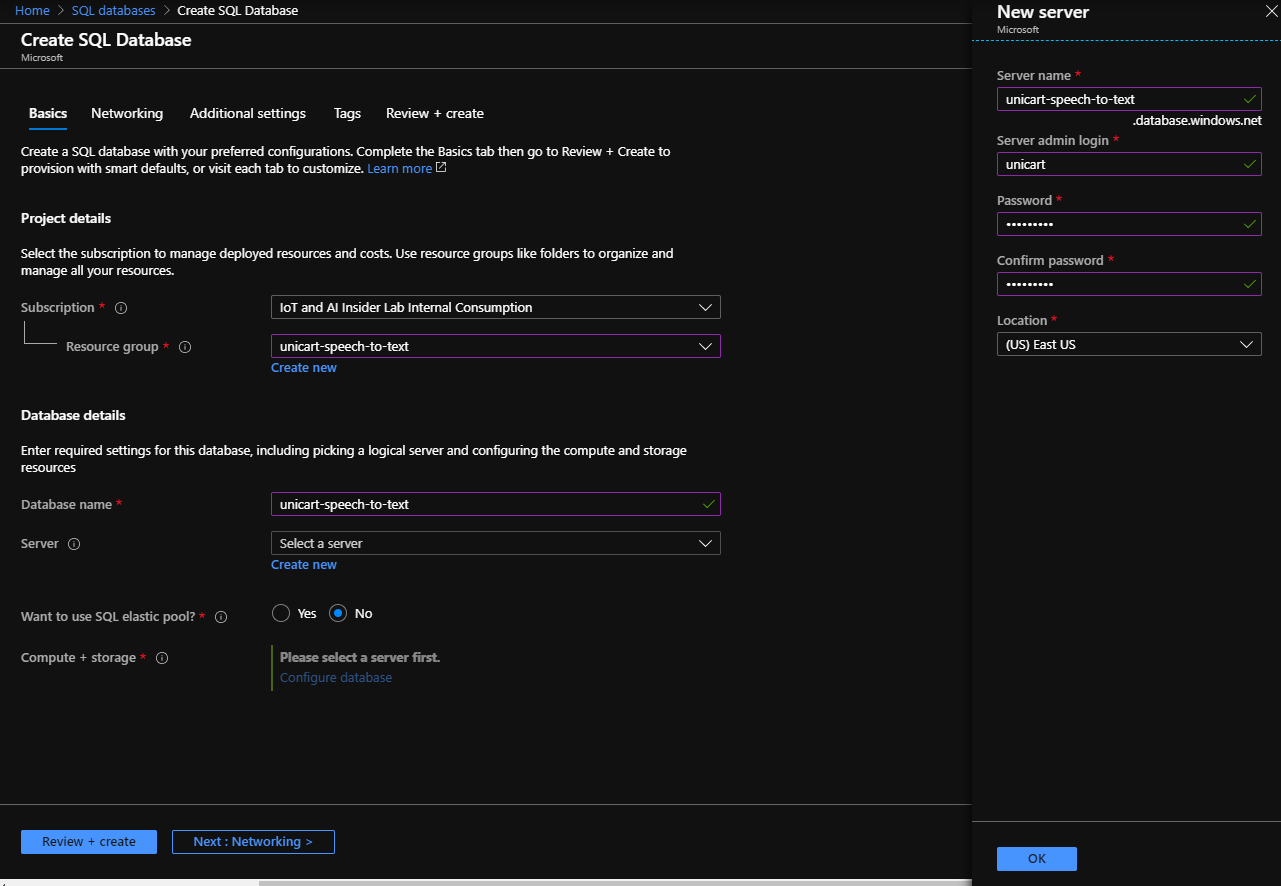
Choose name, subscription, location, and pricing tier. Make sure to choose the previously created resource group.

A screenshot of a cell phone

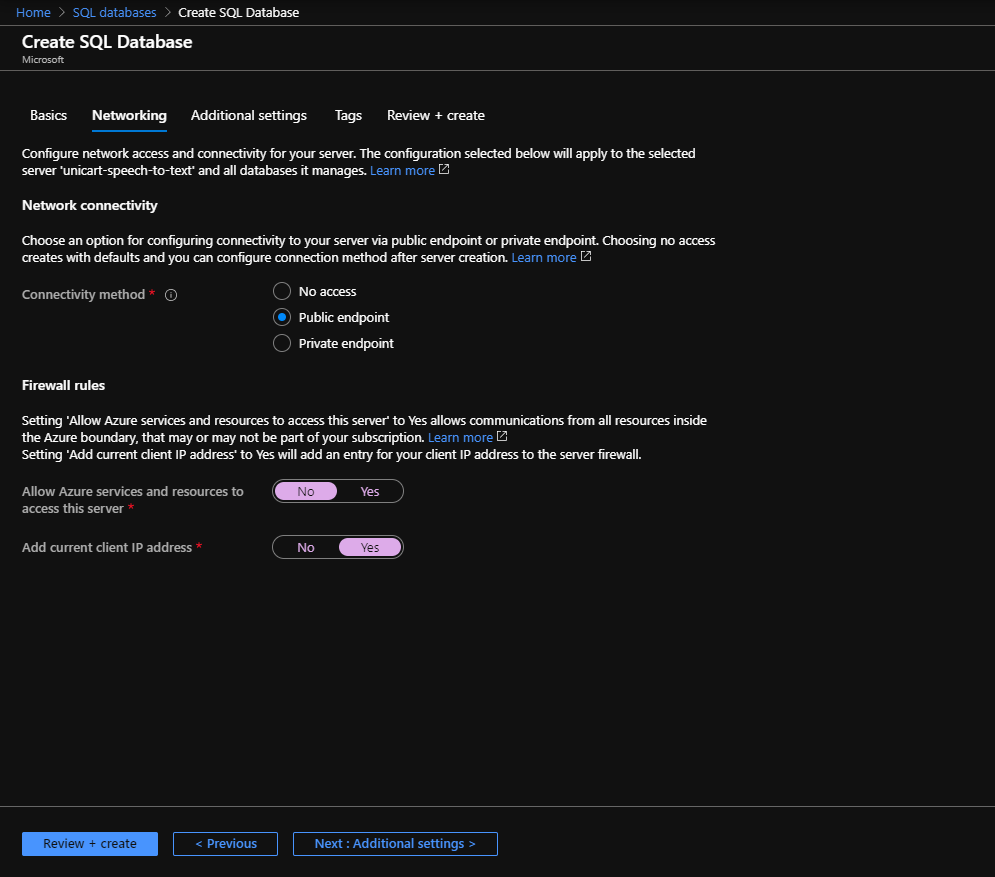
Description automatically generated

**Step 3**: Create a SQL Database and Server

When creating the database, choose “Create new” when selecting a SQL Server. A blade will open on the right to enter information to create the server such as the server’s name, admin login, password, and location.



After entering this information, choose the desired level of “Compute + storage” then go to the next “Networking” step. Choose “Public endpoint” and select “Yes” for “Add current client IP address”.



Click through to “Additional settings”, leave all of the “Additional settings” as their default.

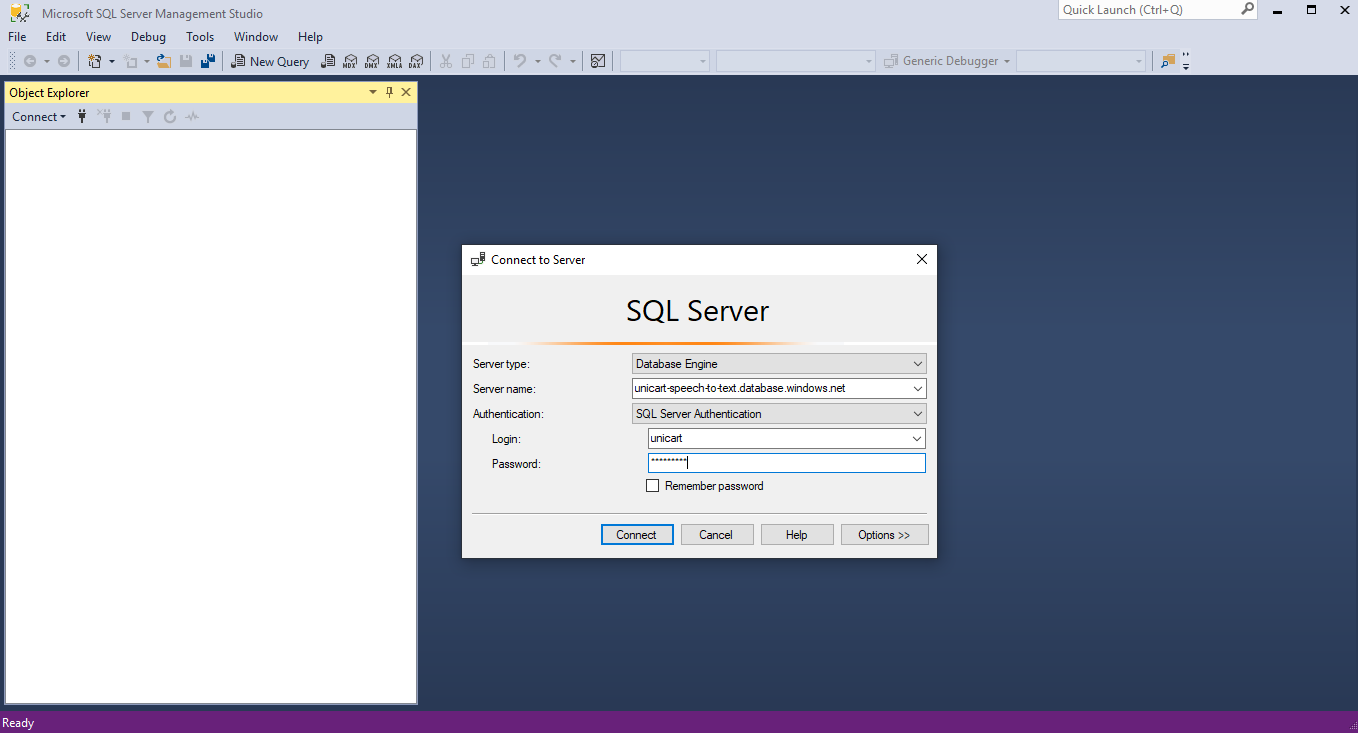
A screenshot of a cell phone

Description automatically generated

Click through to the “Tabs” then “Review + create”. Click “Create”.

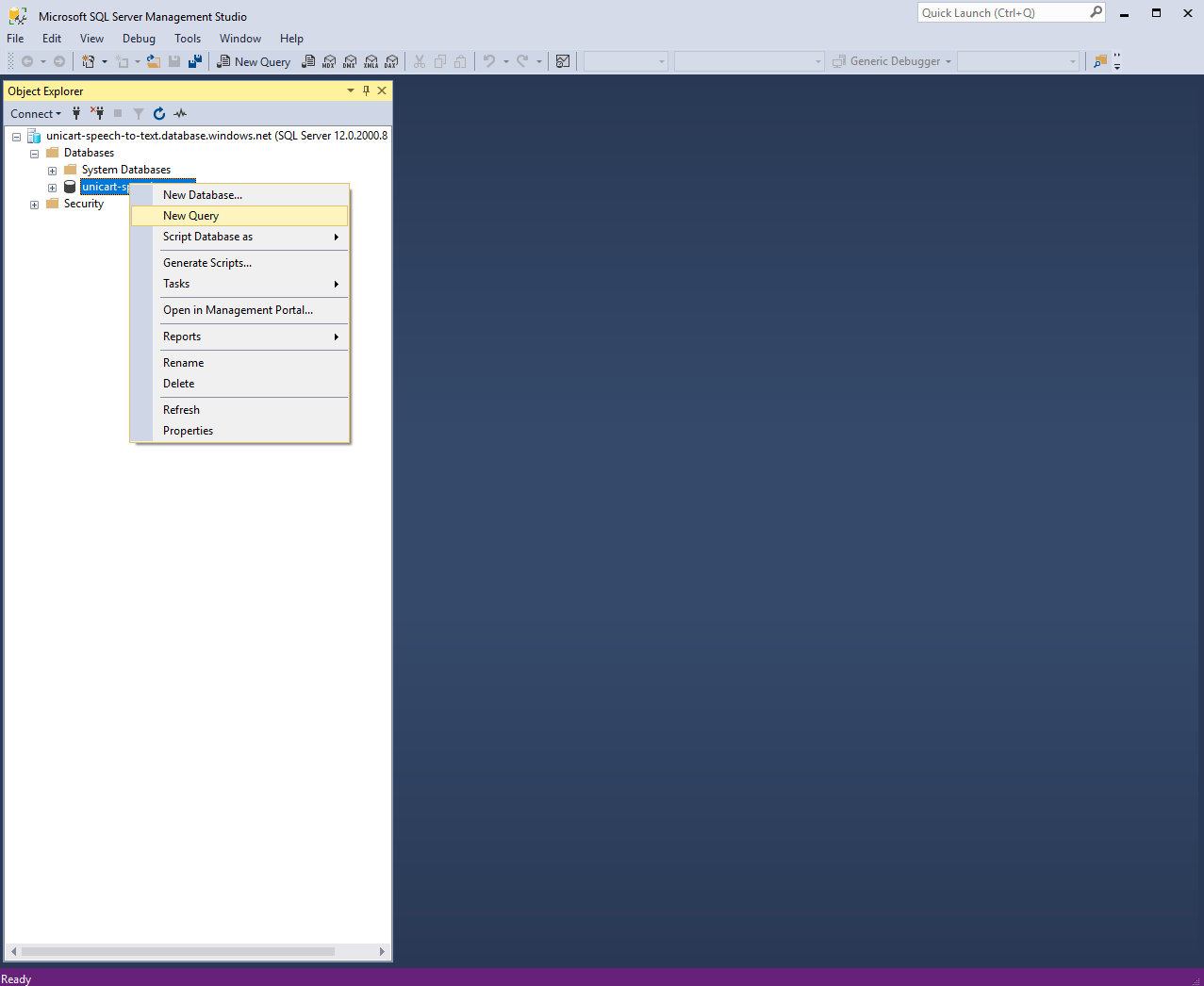
**Step 4**: Connect Microsoft SQL Server Management Studio (SSMS) to newly created database

Download and install Microsoft SQL Server Management Studio (SSMS). Open the application and enter the login info for the database.



**Step 5**: Create a table in the database

Expand the “Databases” folder in the SSMS “Object Explorer”. Right click on the name of your database and select “New Query”



Copy and paste the following script into the query window:

CREATE TABLE SPEECH\_TO\_TEXT\_RESULTS

(ID nvarchar(30),

RESULTS nvarchar(255),

TS DATETIME NOT NULL

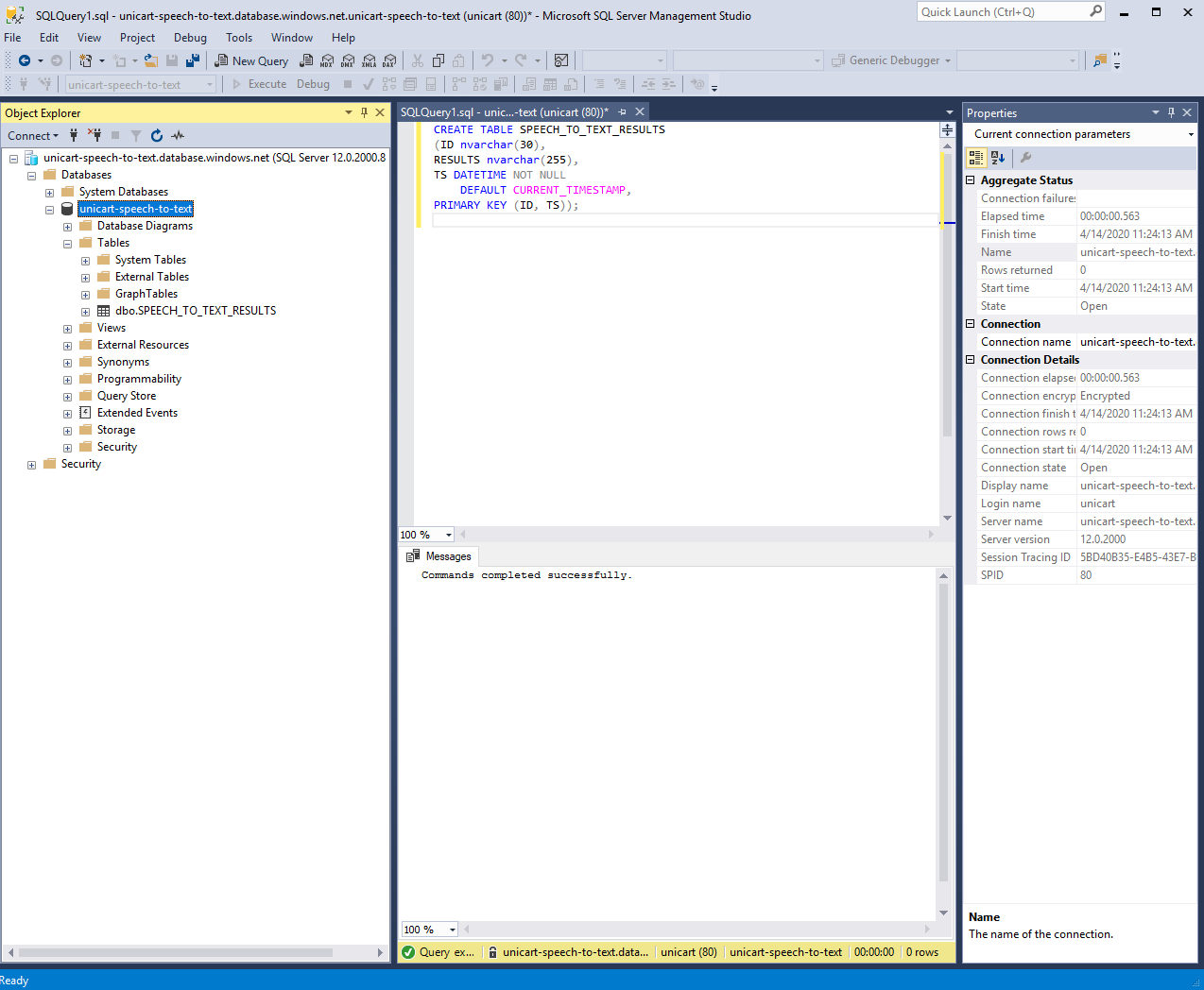
DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (ID, TS));

This will create a table in the database named SPEECH\_TO\_TEXT\_RESULTS. Every entry in the table will have:

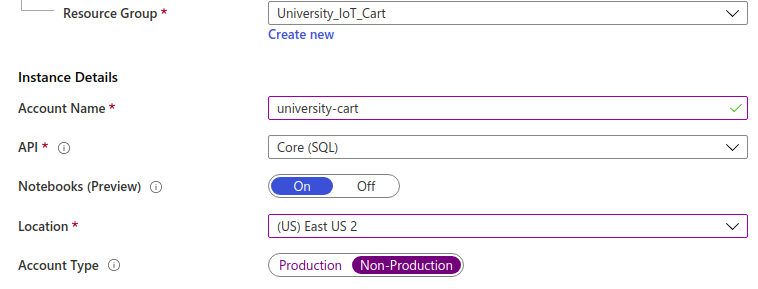
* **ID**: A unique identifier for the device doing the audio recording
* **RESULTS**: The results of the speech to text API call
* **TS**: A timestamp of the current time. This feature will be automatically generated.

Execute the script with the green “Execute” arrow. Expand the database object in the “Object Explorer”, then expand the “Tables” folder. There should now be a table in this folder with the name “dbo.SPEECH\_TO\_TEXT\_RESULTS”.



## Setting up the Raspberry Pi

Step 1: In the Azure portal create a new Cosmos DB. Select Core(SQL) as the API, and enable notebooks.



Step 2: Create and Review

Select Create and Review. Then select Create.

Step 3: Connect to your database

After you create the resource group you will need to create a collection to store the data. Go to <https://cosmos.azure.com>, login, select your subscription, and the Cosmos Account you just created.

Step 4: Create a Database and Collection

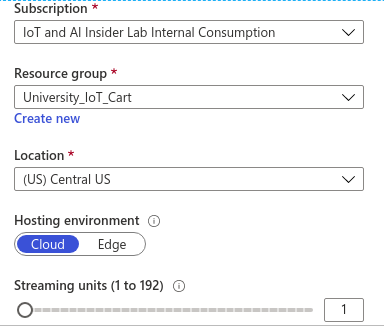
In the GitHub Repo for this project there are 2 iPython notebooks. Go into the My Notebooks section of Cosmos Database and uplaod the 2 iPython Notebooks. Then run the notebook titled Pi\_Environment-Monitor-Setup.ipynb by clicking on the Run command. This notebook will create the database, collection, and user defined function.

## Stream Analytics Job

The next step is to create a Stream Analytics Job. Stream Analytics is a distributed stream processing and aggregation service. We will use it to put data from IoT Hub to Cosmos DB.

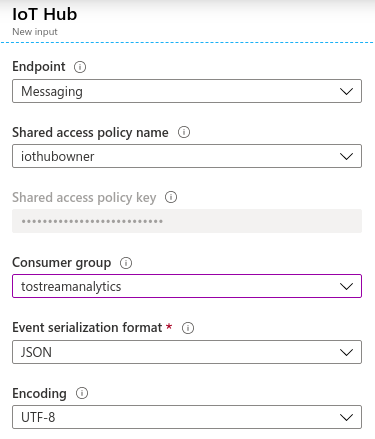
Step 1: Create the Stream Analytics Job

In the Azure Portal create a Stream Analytics Job. From here give it a name, assign it to a resource group, Gie it a



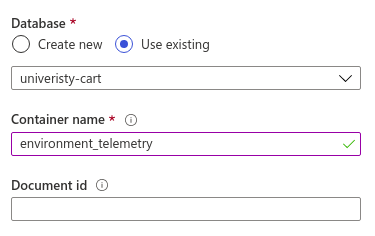
Step 2: Connect to IoT Hub

We are going to add IoT Hub as an input. Do to this we will go to the inputs section in Stream Analytics and select the IoT Hub you created and the consumer group you created for that IoT Hub. For this exercise we are going to ingest the uncompressed JSON telemetery messaging.



Step 3 Output to CosmosDB

Next we are going to add an output from Stream Analytics to CosmosDB. To do this click on output and add a new CosmosDB output. Select the database and collection you created in the previous step. Leave Document id blank.



Step 4: Query

Next select the “Query” in the menu and in the query editor window put in the following query.

**SELECT** GetMetadataPropertyValue(IoTHub, '[IoTHub].[ConnectionDeviceId]') + **cast**( **MONTH**(EventEnqueuedUtcTime) **as** **nvarchar**(**max**)) + **cast**(**Year**(EventEnqueuedUtcTime) **as** **nvarchar**(**max**)) **as** partitionkey  
      , GetMetadataPropertyValue(IoTHub, '[IoTHub].[ConnectionDeviceId]') **as** deviceid  
      ,temperature  
      , humidity  
      , EventEnqueuedUtcTime **as** time  
      ,device\_type  
**INTO**  
    cosmosdb  
**FROM**  
IoTHub   
**WHERE** device\_type = 'pi\_environment\_monitor'

In the code we are creating a parition key so that CosmosDB knows where to store the data. Next we are adding the deviceid, temperature, humidity, and time.

Next select Save.

Step 5: Running the Job

After you have saved your query, go back to the “Overview” page and click Start.

# Getting Ready

Next we are going to use the Rapberry Pi to write and execute the code.

You will need to pip install the IoTHub device client. In a terminal windows:

pip3 **install** azure-iothub-device-**client**

Then you will need to open create a new directory in a command prompt

Mkdir HumidtyIoT  
Cd HumidityIoT

# How to do it

Create a new file called telemetry.py. Raspherry Pi has a code editor called Thorney you can use.

First import the libraries

**import** random  
**import** time  
**import** threading  
from azure.iot.device **import** IoTHubDeviceClient, Message, MethodResponse

Next set the variables. We got the device connection string in the section about creating a device in IoT Hub.

CONNECTION\_STRING = "The device connection stringCONNECTION\_STRING = "The device connection string"  
  
TEMPERATURE = 20.0  
HUMIDITY = 60  
INTERVAL = 1  
readings = {'temperature':0,'humidity':0, 'device\_type':'pi\_environment\_monitor'}

Next, we are going to get the readings from the sensors

def getReadings():  
readings['temperature'] = TEMPERATURE + (random.random() \* 15)  
readings['humidity'] = HUMIDITY + (random.random() \* 20)  
msg\_txt\_formatted = json.dumps(readings)  
message = Message(msg\_txt\_formatted)  
return(message)

Then we are going to have an infinate loop of getting readings and sending it to IoT Hub.

def iothub\_client\_telemetry\_run():  
   
    try:  
        client = IoTHubDeviceClient.create\_from\_connection\_string(CONNECTION\_STRING)  
        print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit" )  
   
        **while** True:  
            message = getReadings()  
   
            # Send the message.  
            print( "Sending message: {}".format(message) )  
            client.send\_message(message)  
            print( "Message sent" )  
            time.sleep(INTERVAL)  
  
    except KeyboardInterrupt:  
        print ( "IoTHubClient sample stopped" )

Finally, we are going to start our program

**if** \_\_name\_\_ == '\_\_main\_\_':   
    print ( "Press Ctrl-C to exit" )   
    **iothub\_client\_telemetry\_run**()

You can find the code for this project listed in the appendix.

# Visualizing Data

Ticket submitted for Cosmos Juypter Notebooks.

# How it works

The DHT 11 temperature and humidity sensor is powered by the Rapberry Pi. The sensor is also connected to the Raspberry Pi through a General Purpose I/O (GPIO). The Raspberry Pi reads the data coming from the sensor and sends it to IoTHub.

# Appendix

Project Code: <https://github.com/Microshak/pi_iot_environment_monitor>

IoT Hub: [Reference](https://docs.microsoft.com/en-us/azure/iot-hub/) and [Tutorial](https://iotschool.microsoft.com/en-us/iot/learning-paths/7zy2NeaeYwsoaCuIueeoSa)

Stream Analytics Job: [Reference](https://docs.microsoft.com/en-us/azure/stream-analytics/) and [Tutorial](https://iotschool.microsoft.com/en-us/iot/learning-paths/6qJe4ohYd2EsAEo6gw6C2G/ILFm6wfcKyeUy2Mk2iAE0?selectedItemId=2G4AQkOJVuAcmuaQiIQ0mm)

CosmosDB: [Reference](https://docs.microsoft.com/en-us/azure/cosmos-db/) and [Tutorial](https://iotschool.microsoft.com/en-us/iot/learning-paths/1z75cTRBNqEA2EigQoQKKe/27PKfzBeLuYCCy6qkA2MaQ)