**Hands-on IoT Lab**

In this tutorial, you configure an IoT sensor device that sends temperature and humidity data to the Remote Monitoring [solution accelerator](https://docs.microsoft.com/en-gb/azure/iot-accelerators/iot-accelerators-what-are-solution-accelerators) and then output that data to a SQL database and a PowerBI dashboard.

*Before you start*

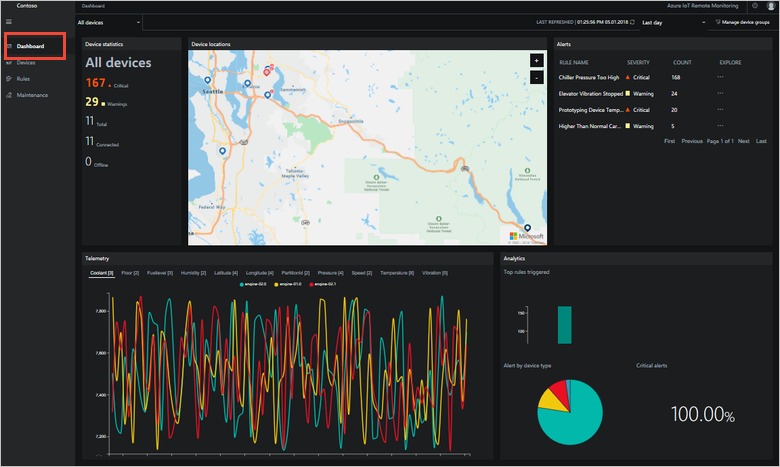
Before you write any code for your device, deploy your Remote Monitoring solution accelerator and add a new physical device to the solution.

*Deploy your Remote Monitoring solution accelerator*

The **MXChip DevKit** device you create in this tutorial sends data to an instance of the Remote Monitoring solution accelerator.

To deploy a Remote Monitoring solution, go to <https://www.azureiotsolutions.com/Accelerators> and log in using an account which has a live Azure subscription. Under Remote Monitoring, click on the Try Now button. Give your Remote Monitoring solution a name, choose your Azure subscription and under Region, choose West Europe. Click the Create Solution button and wait for your Remote Monitoring Solution to deploy.

When the deployment process for the Remote Monitoring solution finishes, click **Launch** to open the solution dashboard in your browser.



*Connect the MXChip DevKit to the Remote Monitoring Solution*

The [MXChip IoT DevKit](https://aka.ms/iot-devkit) is an all-in-one Arduino compatible board with rich peripherals and sensors. You can develop for it using [Visual Studio Code extension for Arduino](https://aka.ms/arduino). And it comes with a growing [projects catalog](https://microsoft.github.io/azure-iot-developer-kit/docs/projects/) to guide you prototype Internet of Things (IoT) solutions that take advantage of Microsoft Azure services.

*Configure the MXChip IoT DevKit device*

1. Plug the MXChip into the PC into the PC with the USB cable. The device should display a green light to show indicate that it is connected.
2. Check the sensors are all working – pressing button B will cycle through the various sensors on the device. The sensors and their readings will be displayed on the little screen.



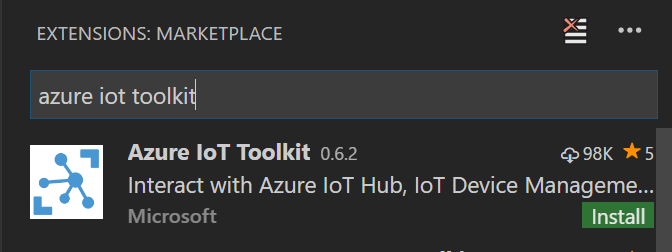
*Prepare the development environment*

1. Download the zip package containing all the tools and packages that you need from here: <https://aka.ms/devkit/prod/installpackage/latest>
2. Locate the zip file and extract the files. Double-click on the Install file to install all the tools. During the installation you will see the progress of each tool or package. It will take about ten minutes in total and will install the following:

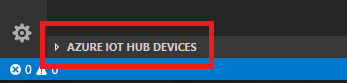
* Node.js and Yarn: Runtime for the setup script and automated tasks.
* Azure CLI 2.0 MSI - Cross-platform command-line experience for managing Azure resources. The MSI contains dependent Python and pip.
* Visual Studio Code (VS Code): Lightweight code editor for DevKit development.
* Visual Studio Code extension for Arduino: Extension that enables Arduino development in Visual Studio Code.
* Arduino IDE: The extension for Arduino relies on this tool.
* MXChip DevKit Board Package: Tool chains, libraries, and projects for the DevKit
* ST-Link Utility: Essential tools and drivers.

Your MXChip DevKit is now set up and ready to go. Disconnect the DevKit from your computer, if it is connected., and start VS Code. You will now need to install the Azure IoT Toolkit for VS Code.

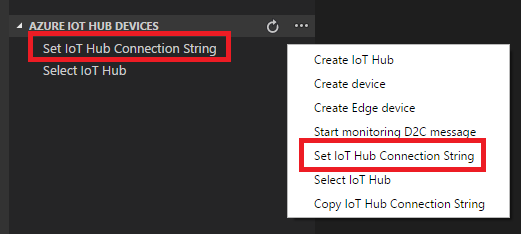
Click on View, Extensions, and type Azure IoT Toolkit. Install the toolkit extension and restart VS Code.



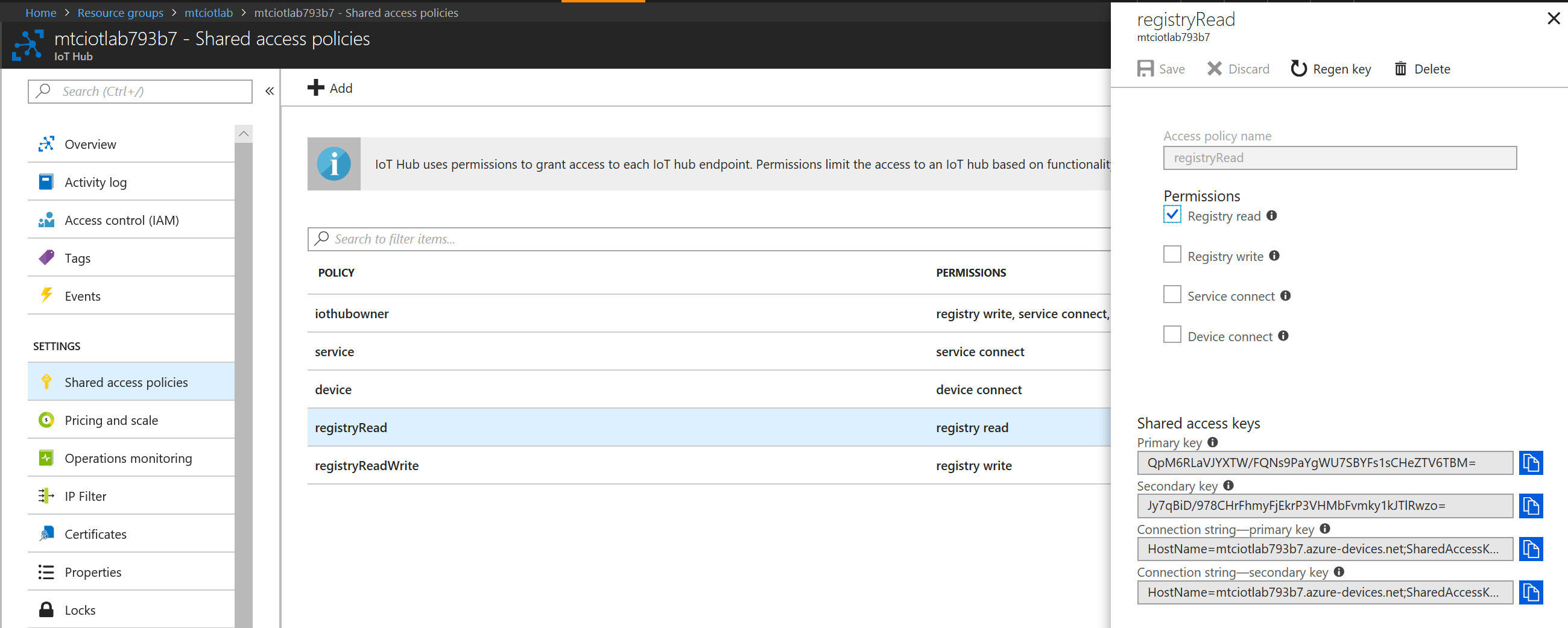
When the toolkit has installed, in VS Code, click "Azure IoT Hub Devices" in the bottom left corner.



Click "Set IoT Hub Connection String" in context menu.



Back in the Azure admin portal (https://portal.azure.com), click on Resource Groups, and then the resource group which contains your Remote Monitoring solution. Click on the IoT Hub, then on Shared Access Policies. Click on the registryRead policy to reveal the shared access keys.



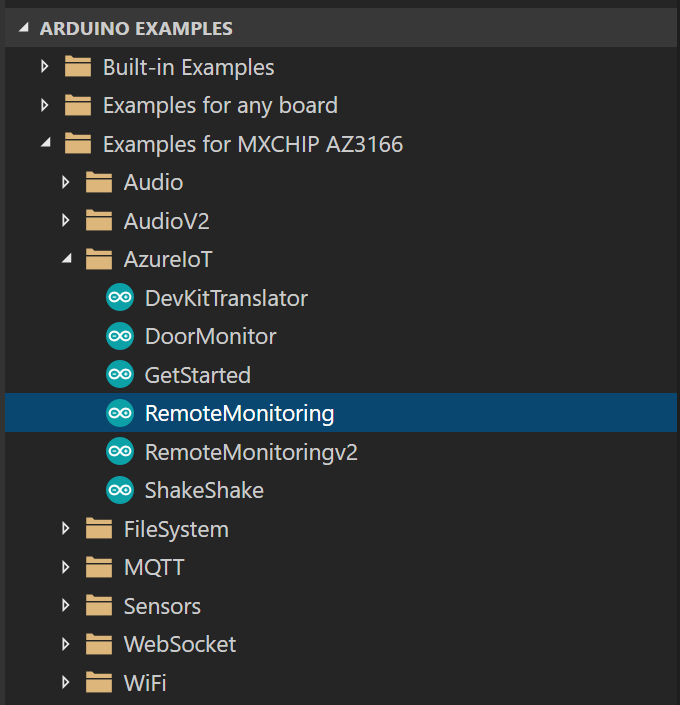
Copy the Connection string - primary key to the clipboard, go back to VS Code and paste it into the Set IoT Hub Connection String box. You should see a list of green devices appear.

Reconnect the DevKit to your computer. VS Code automatically detects your DevKit and opens the following pages:

* The DevKit introduction page.
* Arduino Examples: Hands-on samples to get started with DevKit.

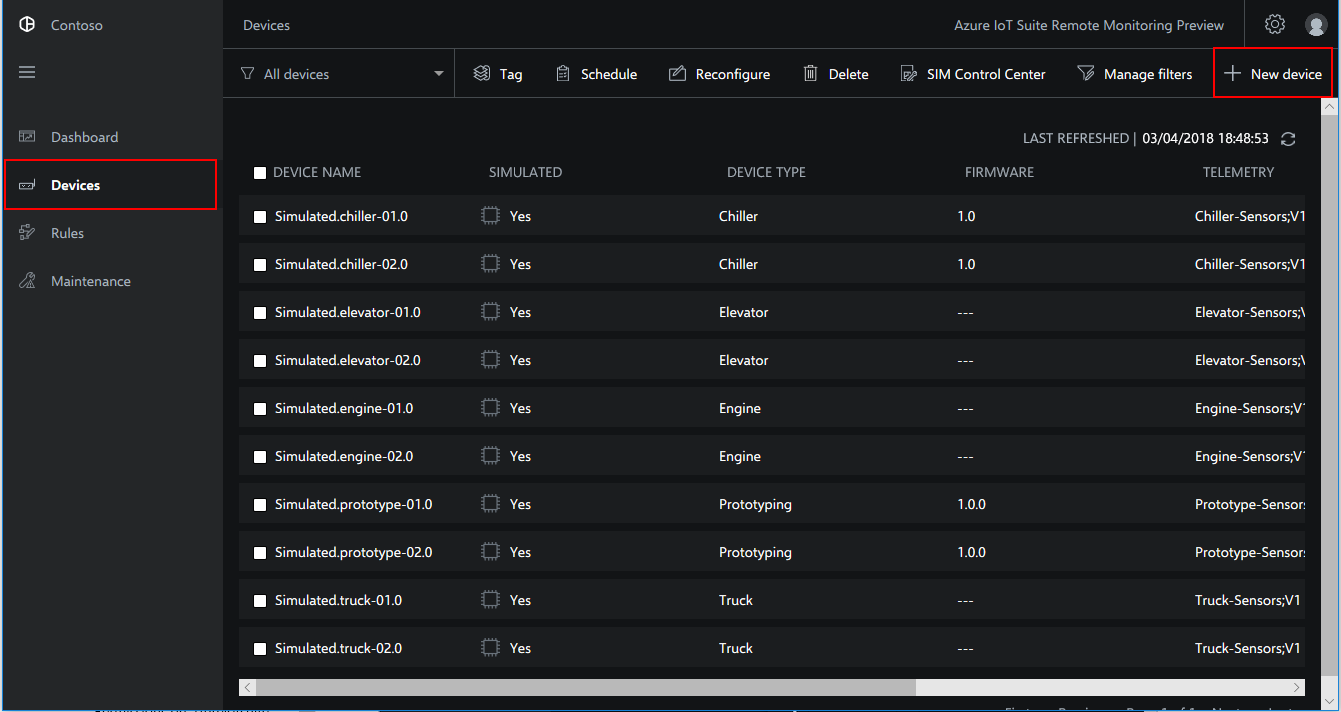
If the Arduino Examples pane doesn’t automatically open, use Ctrl+Shift+P to open the command palette, type **Arduino**, and then find and select **Arduino: Examples**.

Expand left side **ARDUINO EXAMPLES** section, browse to **Examples for MXCHIP AZ3166 > AzureIoT**, and select **RemoteMonitoring (not V2!)**. It opens a new VS Code window with a project folder in it.



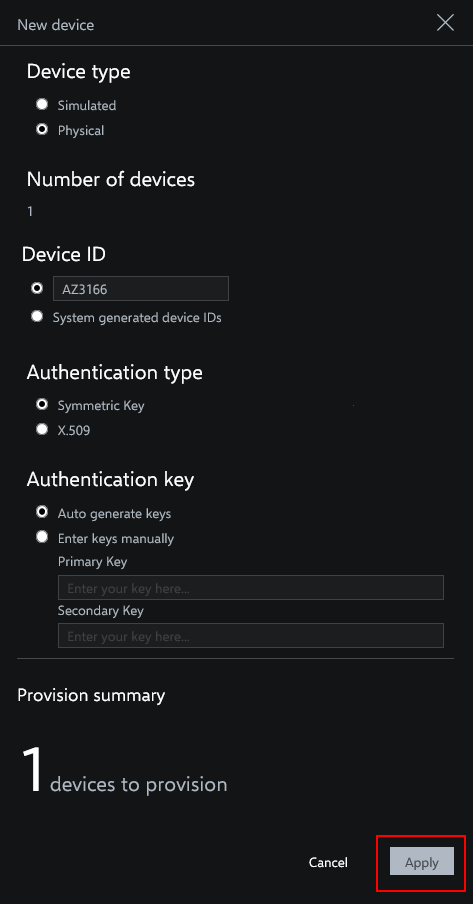
*Add a new physical device*

In the Remote Monitoring Solultion portal, go to **Devices** section and there, click in the **+New Device** button.

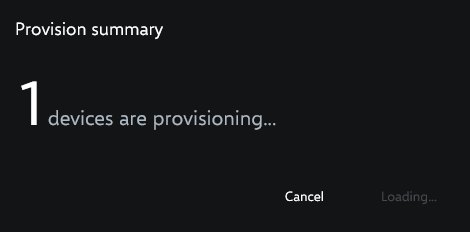


The *new device form* should be filled in.

1. Click **Physical** in the *Device type* section.
2. Define your own Device ID – name your device AZ3166
3. Choose **Auto generate keys** in the *Authentication key* section.
4. Click *Apply* button.



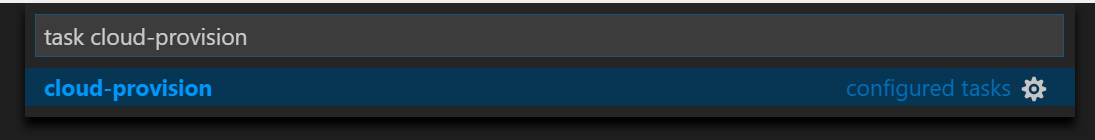
Wait until the portal finishes the provisioning of the new device.



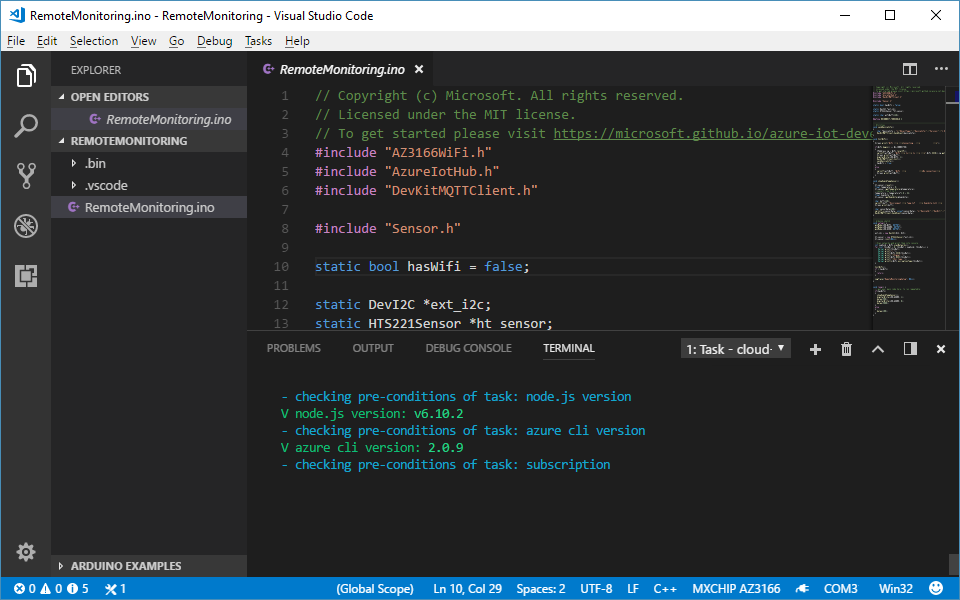
*Build and upload the device code*

Go back to the Visual Studio Code:

1. In the solution window, run your task by pressing Ctrl+P and then type “task cloud-provision” (without the quotation marks) in the box. Press enter.



1. In the VS Code terminal, an interactive command line guides you through provisioning the required Azure services. You will need to pick your subscription, then pick your IoT hub.

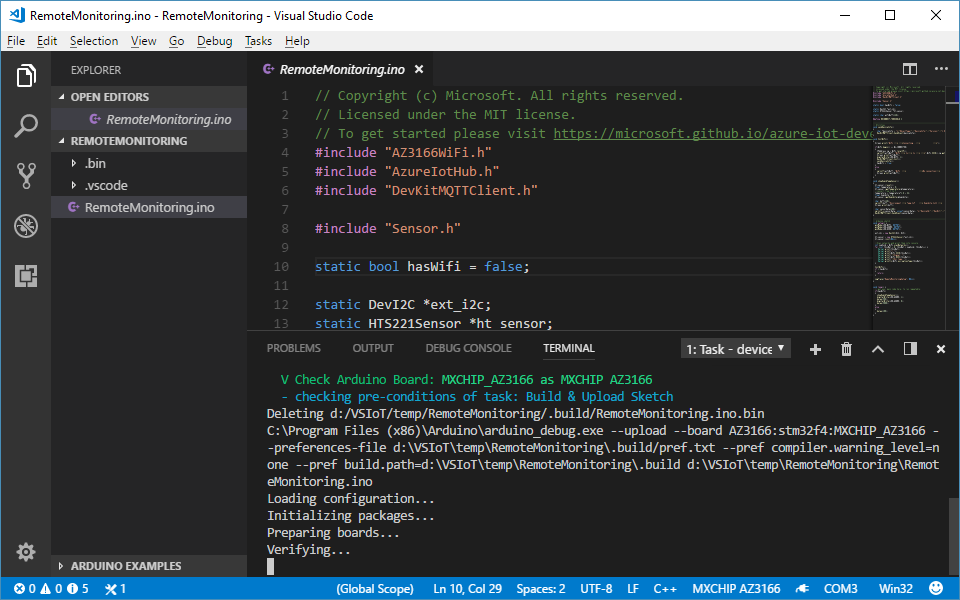


## *Build and upload the device code*

1. Use Ctrl+P to open the command box and type “task device-upload” (without quotation marks) in the box. Press enter.
2. The terminal prompts you to enter configuration mode. To do so, hold down button A, then push and release the reset button. The screen displays the DevKit id and ‘Configuration’.

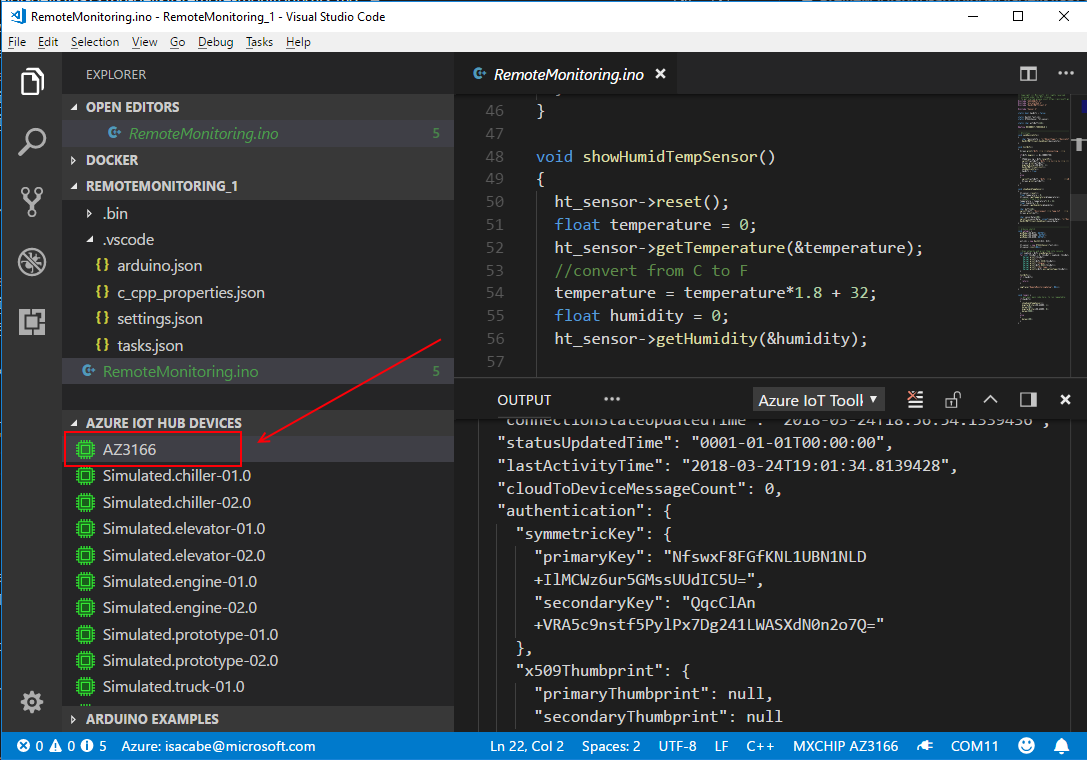
This is to set the connection string that retrieves from task cloud-provision step.

Then VS Code starts verifying and uploading the Arduino sketch:



The DevKit reboots and starts running the code.

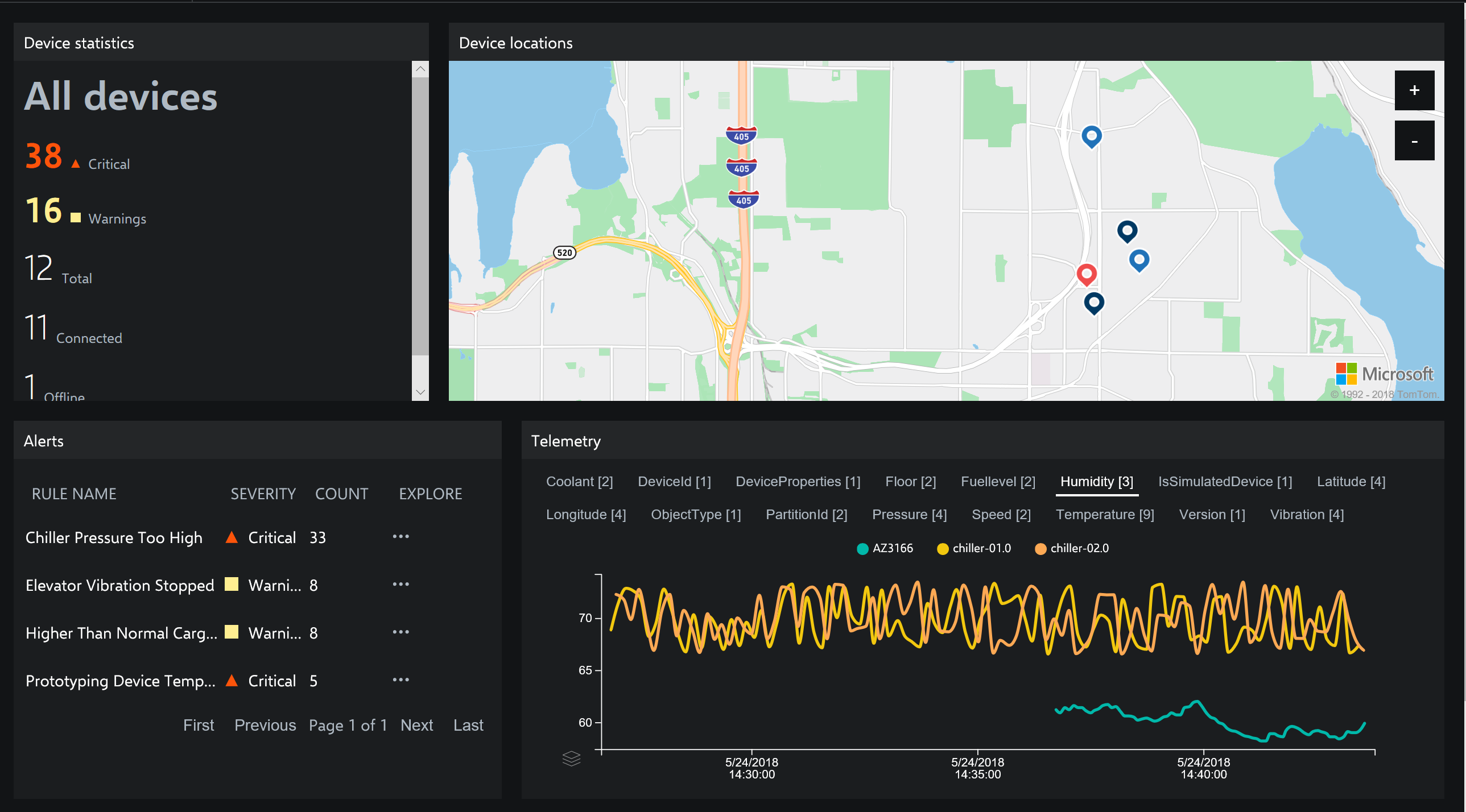
After a few seconds you should see your new physical device turn from blue to green in the VS Code section "Azure IoT Hub Devices". You may need to refresh the window to see it:



*Test the project*

When the sample app runs, DevKit sends sensor data over Wi-Fi to your Azure IoT Suite. To see the result, follow these steps:

1. Go to your Azure IoT Suite, and click **DASHBOARD**.
2. On the Azure IoT Suite solution console, you will see your DevKit sensor status.
3. In the Telemetry section, click on Temperature. You should see your new sensor in there with data flowing into the graph. You may need to refresh the page if you do not see any telemetry.



On the Devices tab, ff you click on the sensor name (AZ3166) a tab opens on the right side of the dashboard, where you can see the MX Chip sensors chart in real time.

*Visualise the data in PowerBI*

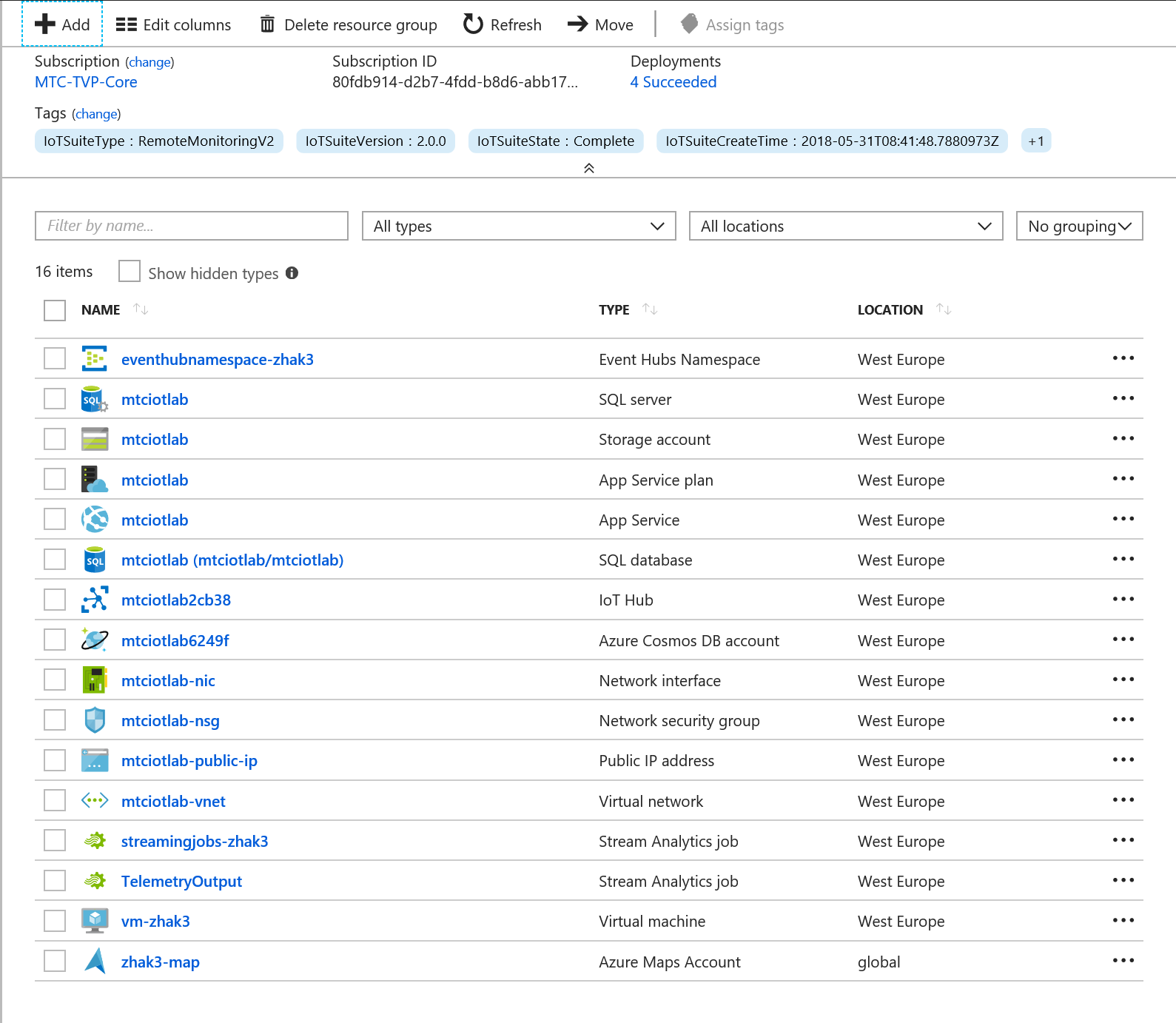
The next tutorial walks you through creating two PowerBI dashboards; one a realtime chart from a live streaming dataset and the other from data sent to a SQL database.

Building dashboards in Power BI means that you can also make each panel interact with one another as you select specific pieces. For example, you could have a filter that shows you only information about your simulated trucks and each piece of your dashboard would interact to show you only simulated truck information. If you would like to use a tool other than Power BI, you can also extend these steps to use your visualization tool of choice and hook into the Cosmos Database, or custom database if you've set one up.

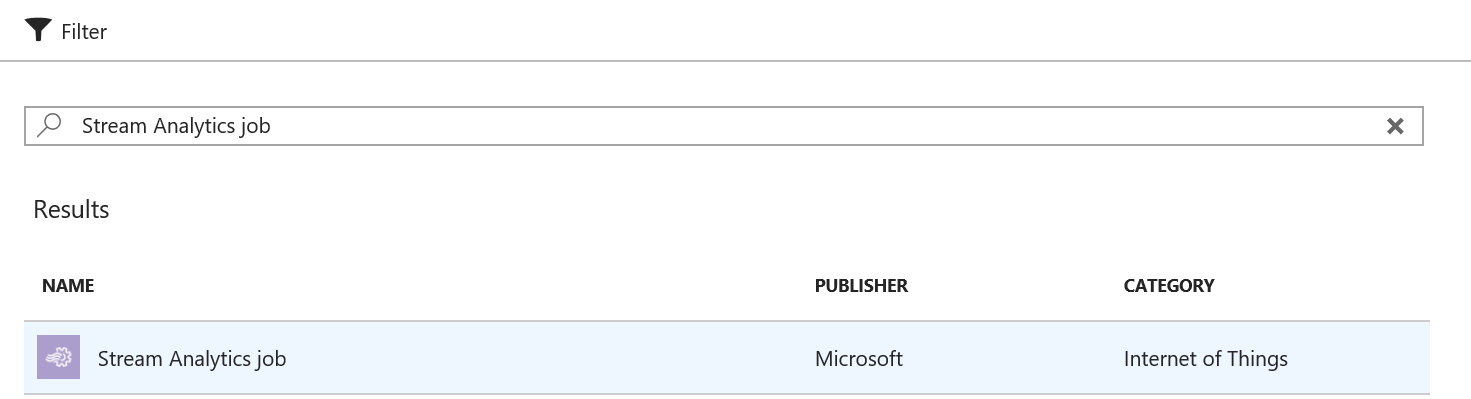
*Create a new Stream Analytics query in the IoT Resource Group*

We want to set up a PowerBi dashboard using a live stream of data from the MXChip and a historic stream from a PaaS SQL database. That is not configured in the IoT hub by default, so we need to create a new Stream Analytics query to output the telemetry to a PowerBI data source and a SQL DB.

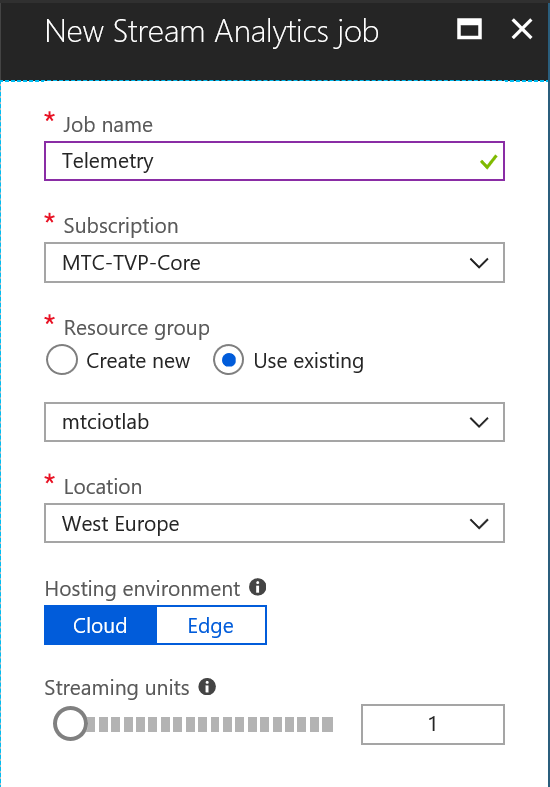
1. Still in the Azure portal, In the left-hand menu, click on Resource Groups (third icon down, picture of a box). Choose your IoT hub resource group and click on it to open the blade.
2. You will see all the resources which make up your IoT hub listed here.



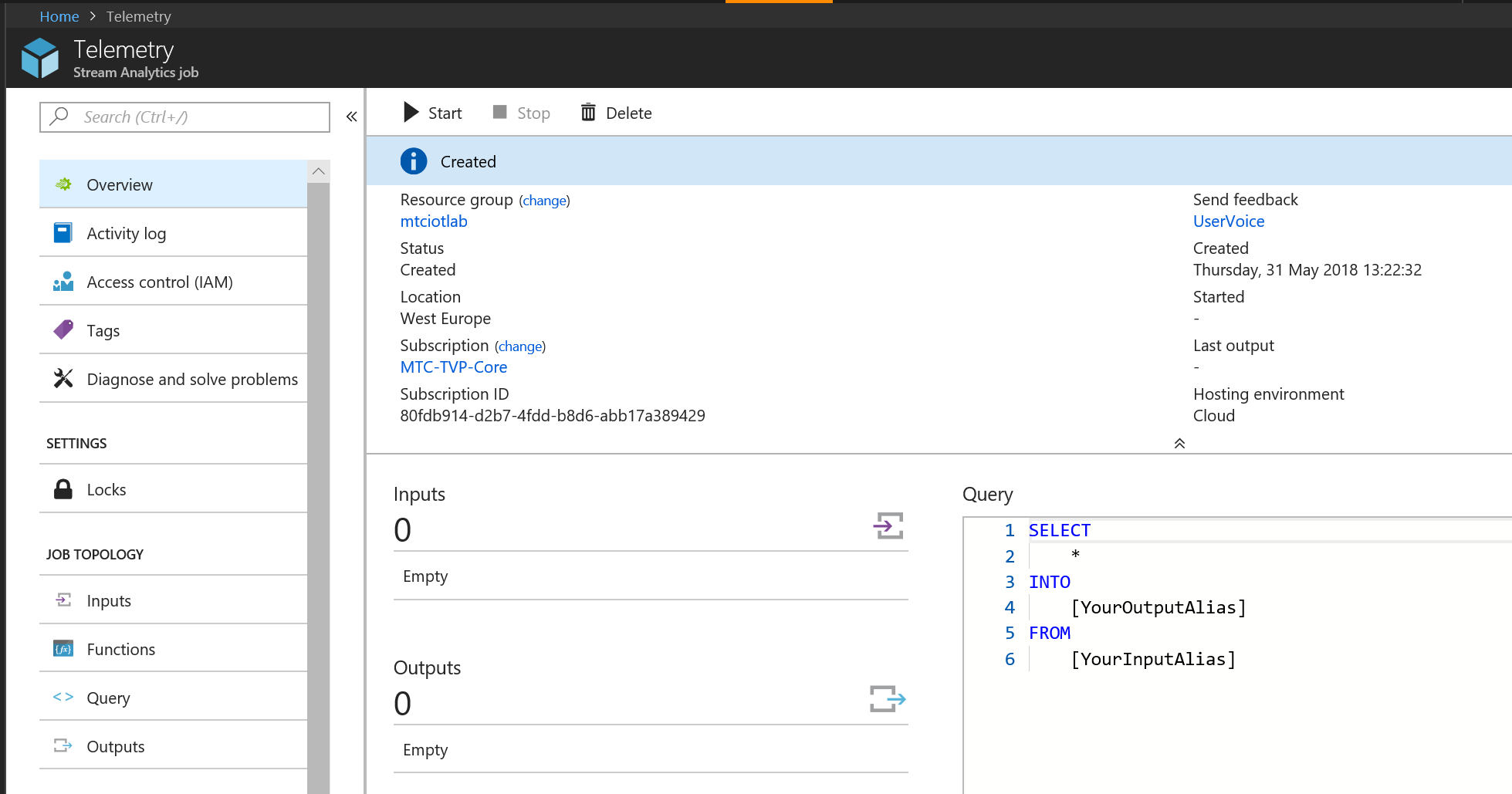
1. Click on the +Add button at the top of the blade. In the Search box, type Stream Analytics. Click on Stream Analytics Job when it appears below, then click on the pink icon to create a new Stream Analytics job. Click the Create button when the blade opens.



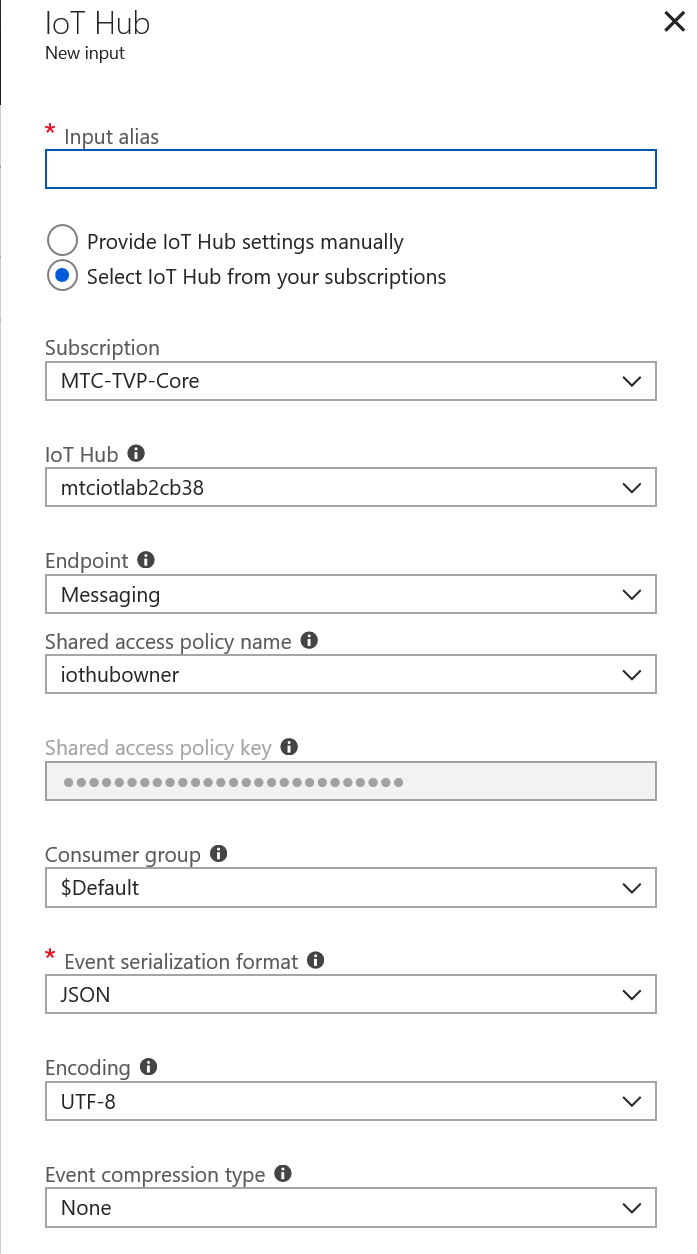
1. Give your new Stream Analytics job a name and choose your subscription and existing Resource Group. Create it in West Europe, leave the Hosting Environment set to Cloud and set the Streaming Units value to 1. Click the Create button when you have completed the boxes.



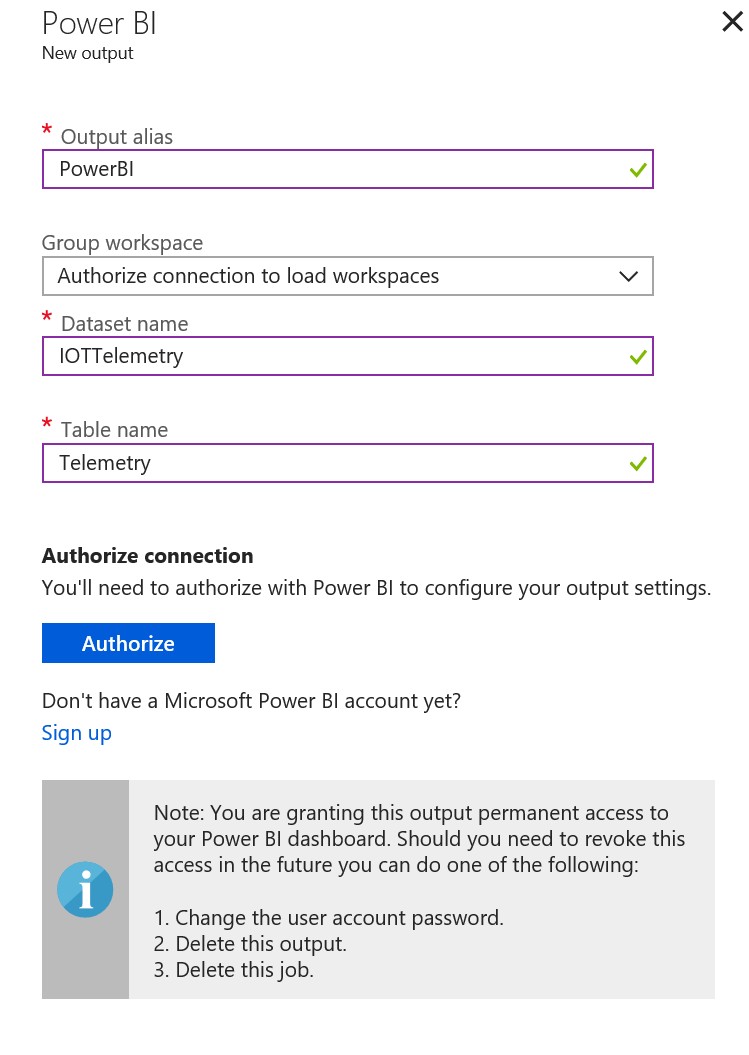
1. When it has created, you should have a screen that looks like this:



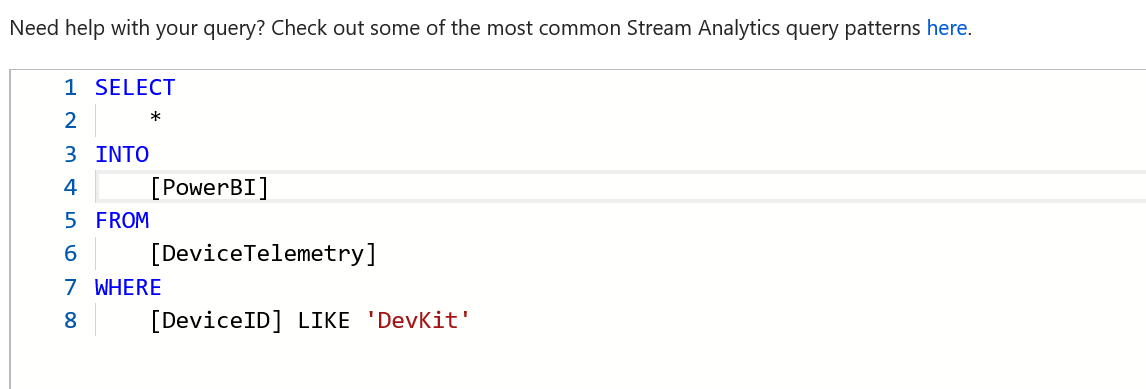
1. You are going to configure an Input first of all, this is where Stream Analytics gets the data from. In our case, it is directly from the IoT Hub. Click on the Inputs option on the left-hand blade menu and click on Add Stream Input. Choose IoT Hub from the options.
2. Give your Input the alias ‘DeviceTelemetry’ and ensure Select IoT Hub from your subscriptions is selected. You only have one IoT Hub, so the rest of the fields should be pre-filled for you with the correct details.



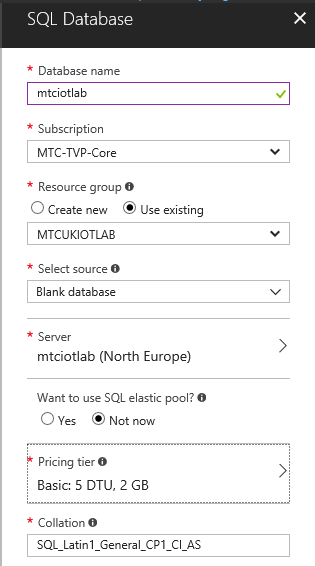
1. Click Save to create your input. Your Stream Analytics job will pull data directly from your devices via this input.
2. Now, firstly, you need to add a PowerBI output to the job. Click on the Outputs box to open the blade.
3. Click on the + sign to add a new output and choose Power BI from the output type list.
4. Enter an alias for your output, to keep things simple call it PowerBi. Enter a Dataset name and a Table name, then click the Authorize button. You will need authorise using an account with a PowerBI subscription, or alternatively, sign up for a free one using the link. This requires a custom company email domain rather than your @outlook.com account.



1. Now you have your first output, you can go back to the Stream Analytics query and direct some data into it. Close the Outputs blade to return to the main SA blade.
2. Now click on the Query menu item. A window will open for you to type your query into. In the main window, enter the following query:

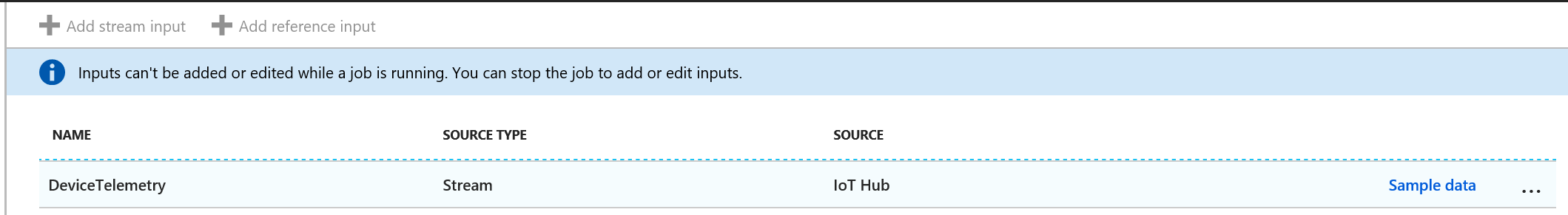


1. This is selecting all messages (\*) from the DeviceTelemetry stream where the Device ID is DevKit, and directing them into your new PowerBI output. If you gave your PowerBI output an alias different to PowerBi, ensure that you use your alias in the INTO field.
2. Once the query has been amended, click Save to save your changes, then close the Query blade. This will take you back to the Stream Analytics blade. Click Start at the top of the blade to start the processing. Click Now in the Job Output Start Time box and click Start to start the query. It will take a few minutes to start up. When it has started, you will see the status change from Starting to Running.
3. Now data will be sent from the MXChip directly to a live streaming feed in PowerBI.
4. Now create a SQL DB for the historic data. In the portal, click on the **+ Create a Resource** in the top left hand of the portal. In the search box, type SQL Database and choose the SQL Database option that it finds. Click on SQL Database at the top of the list, then click the Create button.
5. Give your SQL database a name and choose the existing Resource Group for it. The Select Source drop-down should be left to Blank database.
6. Click the Server box and click the Create a new server option at the top. Give your SQL server a name, and enter a username and password for it. Leave all other options as they are.
7. Click on the Pricing Tier box. Choose the Basic tier. Click Apply, leave the Collation setting as it is and click Create.

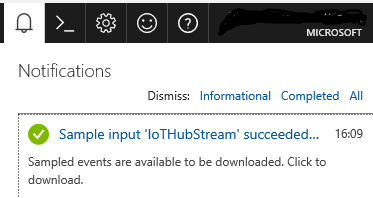


The SQL instance will now deploy. It will take a few minutes.

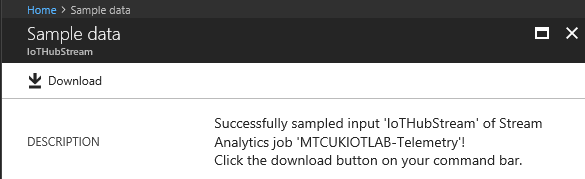
1. You will need to define the database schema to match up with the data coming from the sensor. Firstly, take a sample of the data coming from the sensor so you know what schema you need. To do this, return to the Resource Group and click on the Stream Analytics job which you created earlier.
2. Click on the DeviceTelemetry input. Click the Sample Data link.



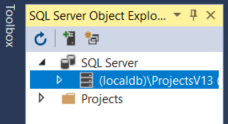
1. Click the Sample button to take a sample of the data being generated. Wait for a minute or so, then click on the bell icon in the top right corner of the portal.



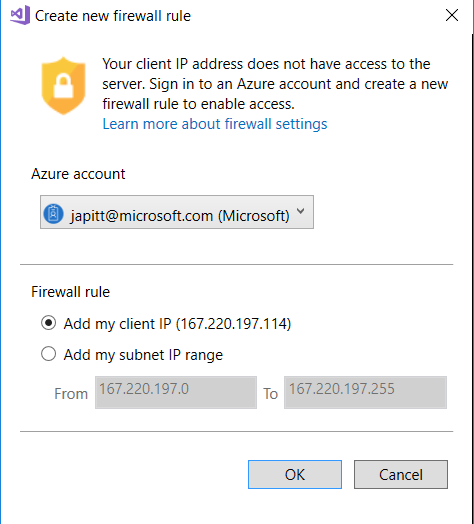
1. You will see that the data sample has been collected. Click on the message to open it, then click on Download to download it.



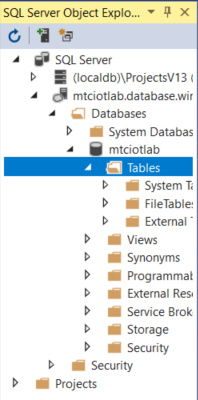
1. You can open the output in Notepad to view it. You will see the format of the messages coming from the device and into the Stream Analytics job. This is the schema that needs to be replicated for the SQL database.
2. In order to create a new database for the message data, open Visual Studio 2017 (the full version, not VS Code) on the training PC. Ensure that you log into Visual Studio using the same account that you used to set up your Azure subscription.
3. Click on the View menu, then click SQL Server Object Explorer. This will open SSOE on the left hand side. Click on the Add SQL Server button.



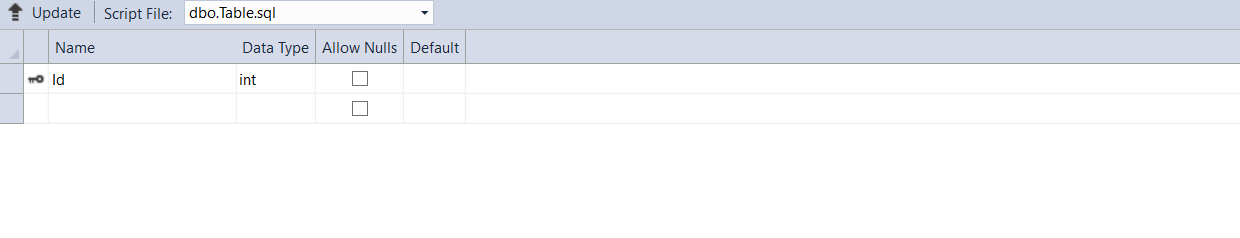
1. In the list of Connect options, choose Azure. This should expand your Azure subscription and show you the database you have just created. Click it in the list and in the boxes beneath, enter the username and password you set up when you created the SQL database. Check the ‘Remember password’ box and click Connect.
2. You may get a message asking you to add a firewall rule to your database to allow client access. Just click OK, Visual Studio will do this for you automatically.



1. Your SQL server should appear in the list. Expand the server, then databases, then click on the name of the database that you specified when you created your database.



1. Right click on the Tables folder and click Add New Table. This will open the Table Designer screen, which looks like this:



1. Follow these instructions exactly! You need to define your table schema in line with the data that is coming from the sensor.

The first line will contain the field name id and will have a little key beside it. Click into the box and change the field name to DeviceID. Change the Data Type in the drop-down to nvarchar(MAX). This should be the data type for all of the fields. Continue down the list to create fields with exactly the names as follows:

DeviceID nvarchar(MAX)

Temperature decimal(5,2)

Humidity decimal(5,2)

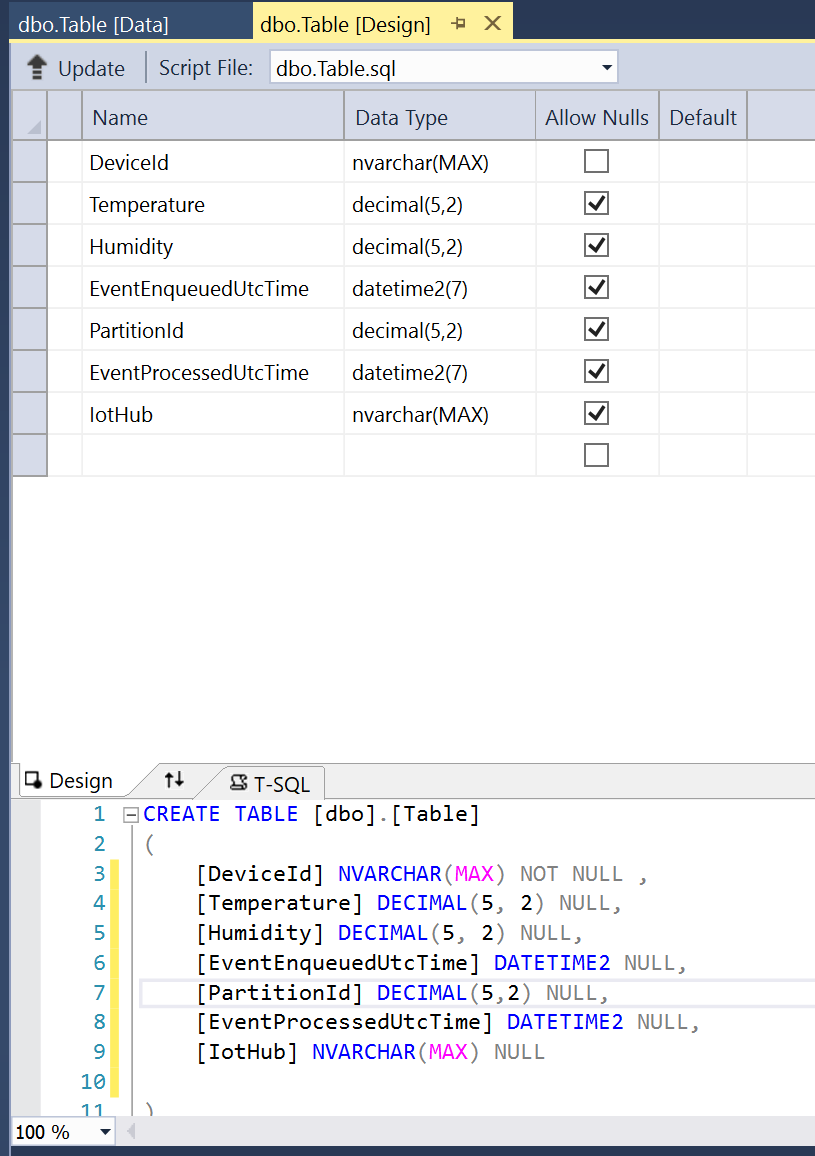
EventEnqueuedUtcTime datetime2(7)

PartitionId decimal(5,2)

EventProcessedUtcTime datetime2(7)

IoTHub nvarchar(MAX)

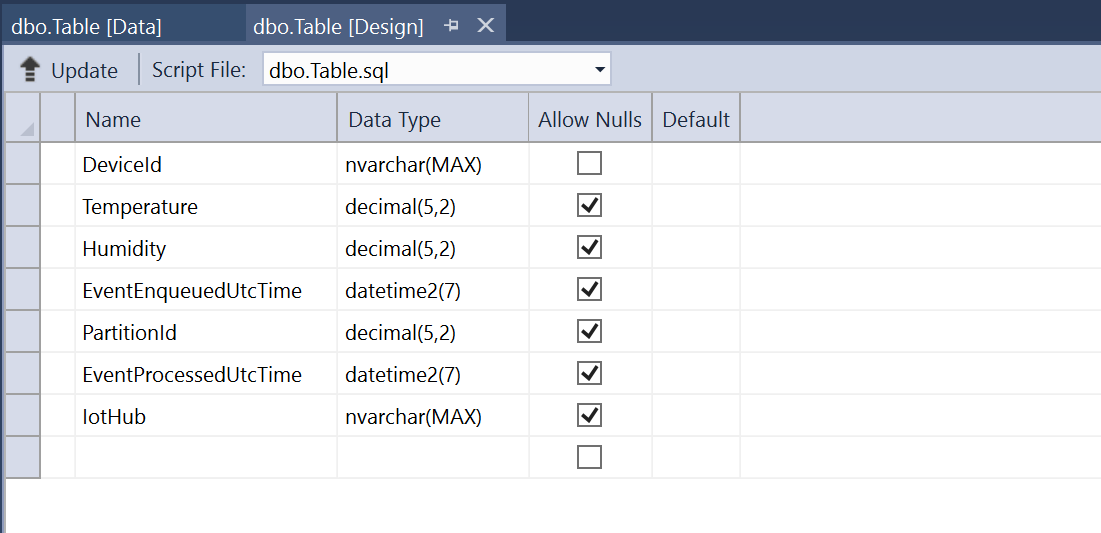
When you change the value to decimal, the numbers in brackets will be (18,0). These numbers represent the length of the value before and after the decimal point. You change this number in the text window beneath the table, just here:



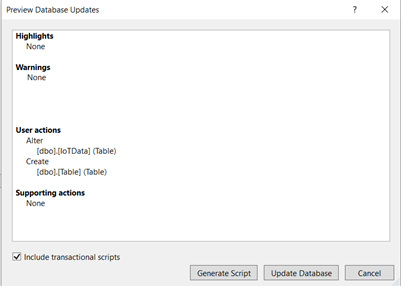
It will just say DECIMAL with no bracket. Add the (5,2) after the word DECIMAL in the T-SQL window and it will update the drop down in the table above.

When you have created all of your fields, right-click on the DeviceID line and click Remove Primary Key. The little picture of the key should disappear.

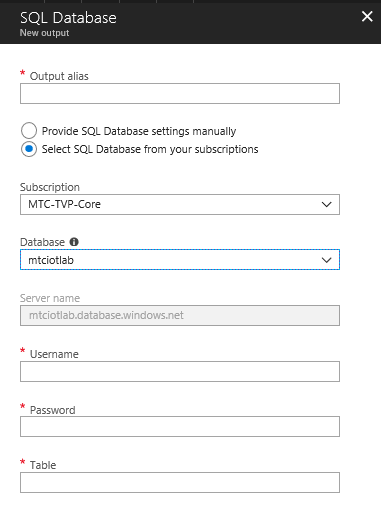
When you’ve finished, your table design should look **exactly** as below:



1. When you have completed your table design, click the Update button in the top left. This generates a script to configure your table. You will see the box below:



1. Click the Update Database button to commit your changes and create your table. The script will run and you should see a success message in the very bottom pane of the window.
2. Now you have created your database, you need to set it up as an output in Stream Analytics. Head back to the Azure portal to the Stream Analytics query. First, you need to stop the Stream Analytics job again. Wait for the job to stop, then click on Outputs.
3. Click the **+** sign to add an output.
4. Pick SQL Database from the list. When the New output blade opens, enter a name for your output alias. Call it Database.



1. Leave the radio button on Select SQL database from your subscriptions, and it should auto-detect the server name that you created earlier. Make sure you pick the database you created from the Database drop-down. Enter the username and password that you assigned to your database when you created it. In the Table box, enter dbo.Table as the table name.
2. Click Create to create and test your new output. Close the output tab to return to the Stream Analytics blade and click on the Query box to amend your query again. Add the following lines to your query **exactly** as they appear below:

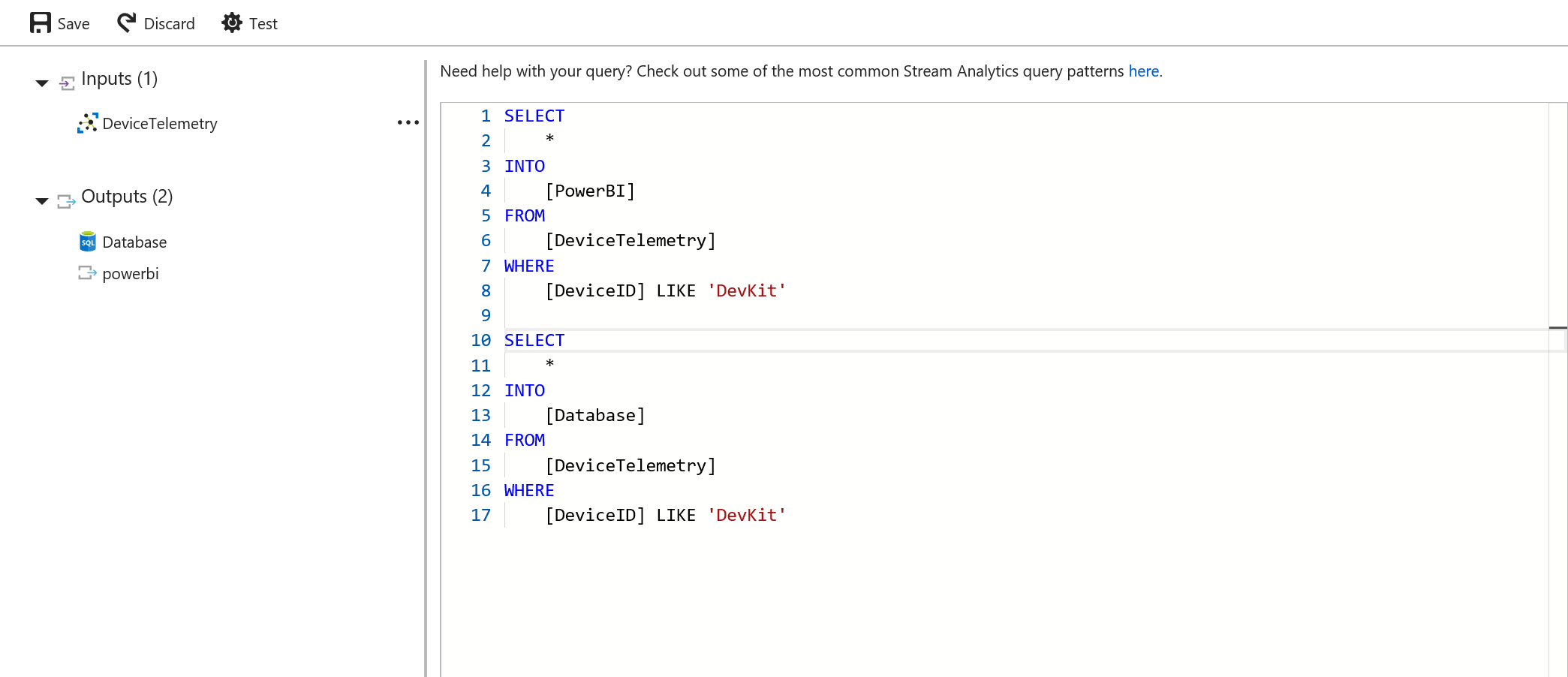
**SELECT** \*

**INTO** [Database]

**FROM** [DeviceTelemetry]

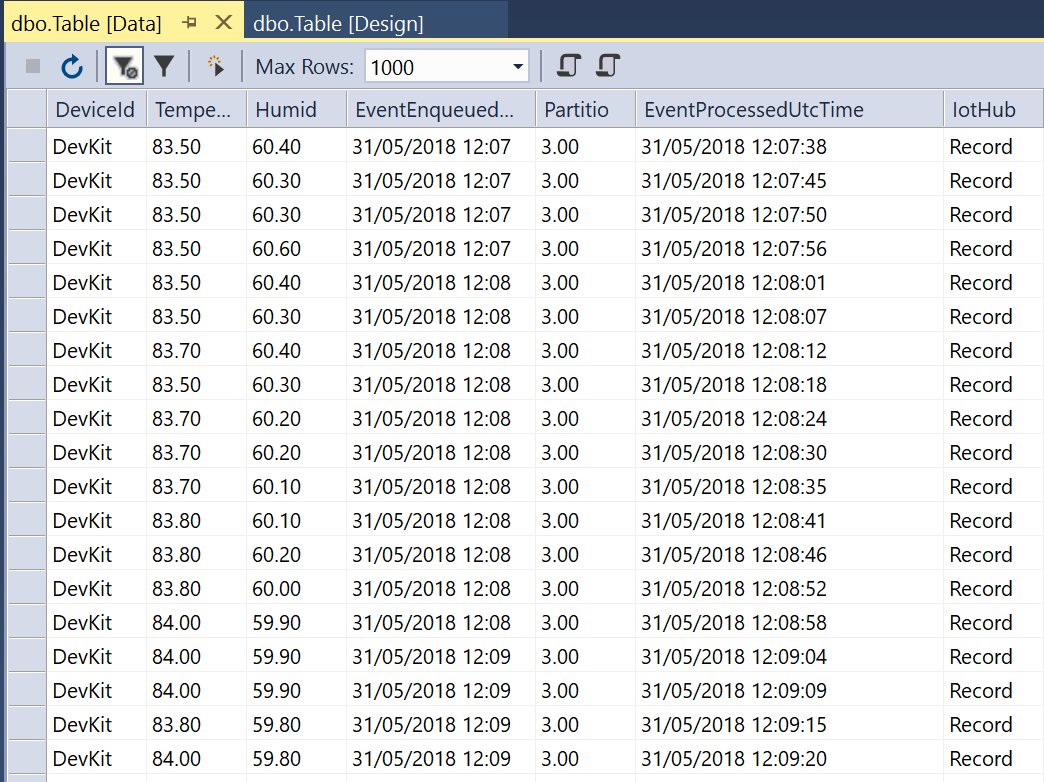
**WHERE** [DeviceId] LIKE ‘DevKit’

1. Your query should now have two new sections and look like this:



1. Click Save to update your query, then click Overview. Click Start to start your Stream Analytics job again. This will take a few minutes.
2. When the Stream Analytics job is showing as Running, we can check that data is reaching your SQL database. Return to Visual Studio, click View, then click SQL Server Object Explorer. You should see your new table listed on the left hand side, dbo.table.
3. Right-click on your table and click View Data. This will open your table in Data view. You see some rows populated with data. If not, click the Refresh button a couple of times, it can take a few minutes for sensor data to reach the table.

This is how your table should look:



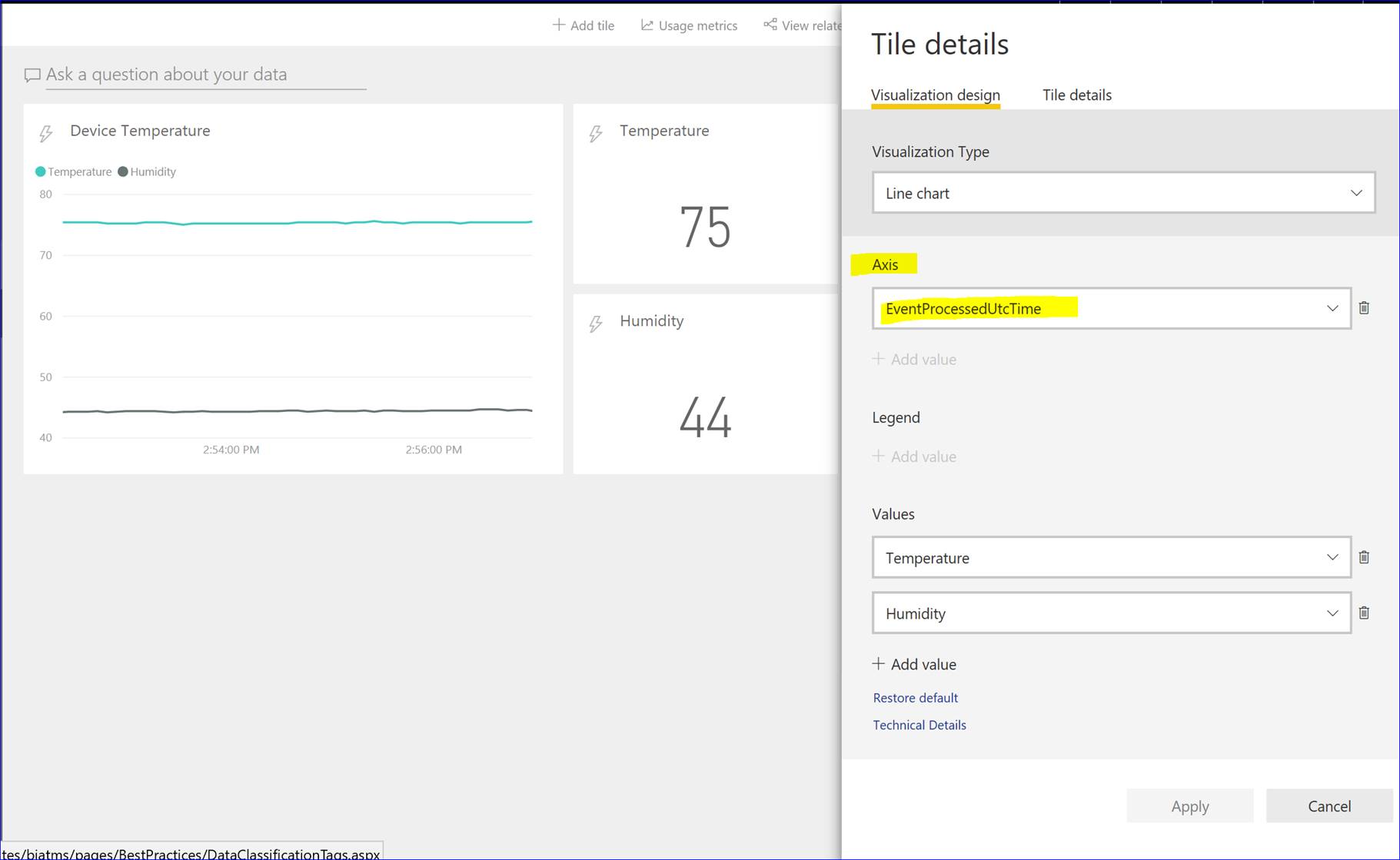
*Set up the PowerBI dashboard*

Now we can display the live and historical temperature data coming from the sensor on a PowerBI dashboard. First we will display the live date online. Open [www.powerbi.com](http://www.powerbi.com) and sign in using the account you used earlier to authorise the Stream Analytics output.

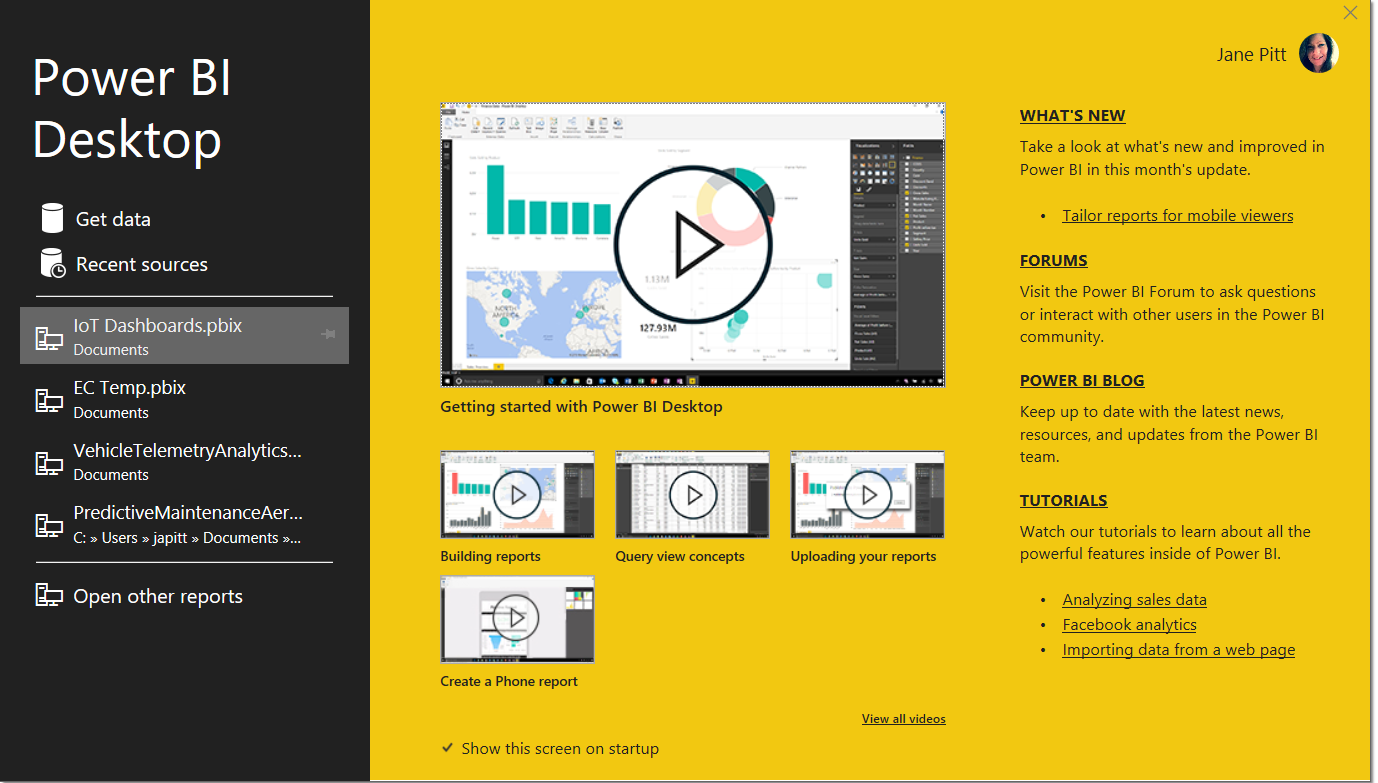
1. In the PowerBI window, click on the three lines button at the top of the left-hand menu.



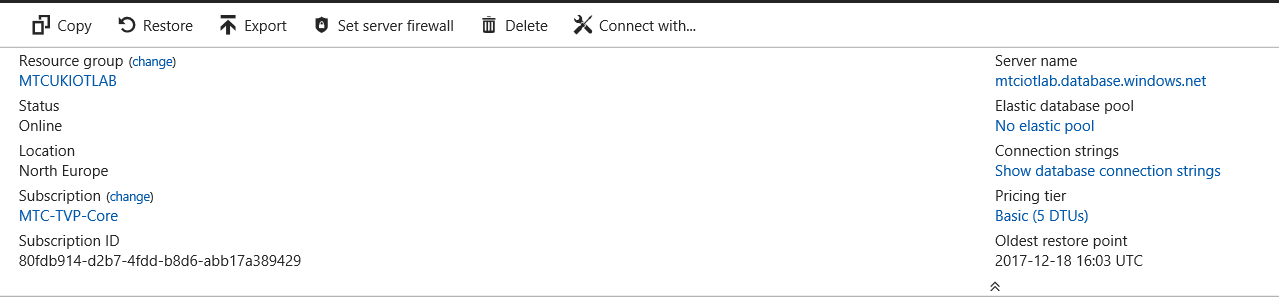
1. Click on My Workspace. In the main window, click on Datasets.
2. You should see listed the Output you created from Stream Analytics listed in your available Datasets. It can take a few minutes to come through, so if it’s not there, wait a few more minutes and refresh the page.
3. When the new Dataset has appeared, in the top right of the window, click the + Create button, and choose Dashboard.
4. Type in a name for your dashboard and click Create.
5. When the dashboard has created, click the Add Tile button from the menu at the top of the screen. In the Select Source window, choose Real-Time Data, Custom Streaming Source. Click Next.
6. Choose your new Dataset from the list. Click Next.
7. In the Visualization Type drop down, choose Line Chart. Add an Axis with the value EventProcessedUtcTime. In the Values drop down, choose Temperature and Humidity. In the Time Window to Display fields, choose 1 minute. click Next.



1. Give your chart a title and click Apply.
2. Add another tile, this time a card. Choose Temperature
3. You will now have a scrolling graph and a box displaying the current temperature. Add some more tiles; a Humidity card, for example. Drag them around to make a neat dashboard. Make the values change by breathing on the sensor.
4. In order to display the data from SQL, you need to use PowerBI Desktop. You can download it from here: <https://powerbi.microsoft.com/en-us/desktop/>
5. Download and install it, it’s free to use. When you open it for the first time, it will ask you to sign in. Sign in with the account you used for the online version of PowerBI, or if you did not have one, sign in with the account you used for your Azure subscription.
6. On the splash screen, click Get Data.

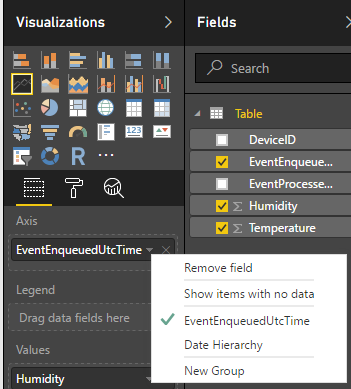


1. Click Azure, then choose Azure SQL Database. Click Connect.
2. You will need your SQL server name. Back in the Azure portal, click on SQL Databases in the left hand menu. It will display your SQL database. Click on it to open the SQL Database blade. This has the name of your SQL server in it. Copy it to the clipboard and paste it into PowerBI desktop.



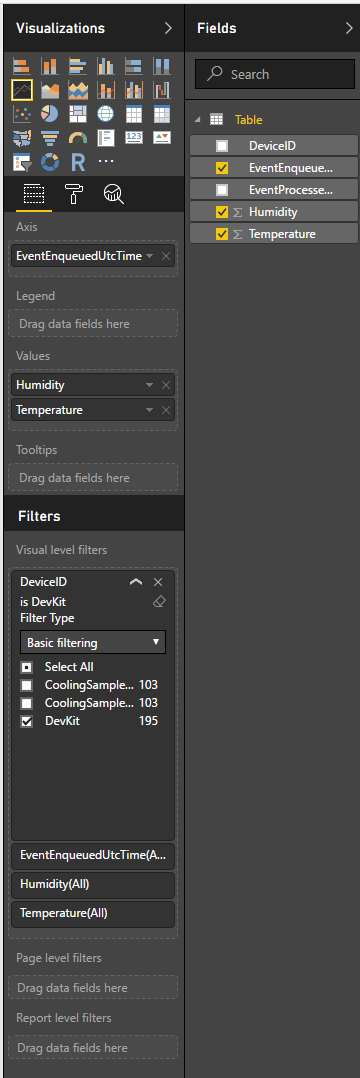
1. You can leave the database name blank. Click OK.
2. You will be asked for the credentials to your database. Ensure you click on the Database tab and enter them in the two boxes, do not enter them on the Windows tab as they will not work!
3. You can either import the data as static data for your report, or connect the SQL database as a DirectQuery, which means you will be able to refresh the data in the report. Choose either.
4. The Navigator should display your SQL server and database. Click on the database to display the tables. You should see your table, called Table, listed. Tick the box beside Table, a sample of the data will display. Click Load.
5. The data is now loaded and ready for you to display in a chart. You can drag fields around in the designer and drop them into the relevant place. Give it a go, it’s easy to manipulate.
6. To draw a line chart plotting history humidity and temperature data over time, choose Line Chart, the first box in the second row down.

* In the Axis box, drag EventEnqueuedUtcTime. Click the tiny down arrow beside the field name and click the option above Date Hierarchy to remove the Day/Year/Month breakdown (if it is present).

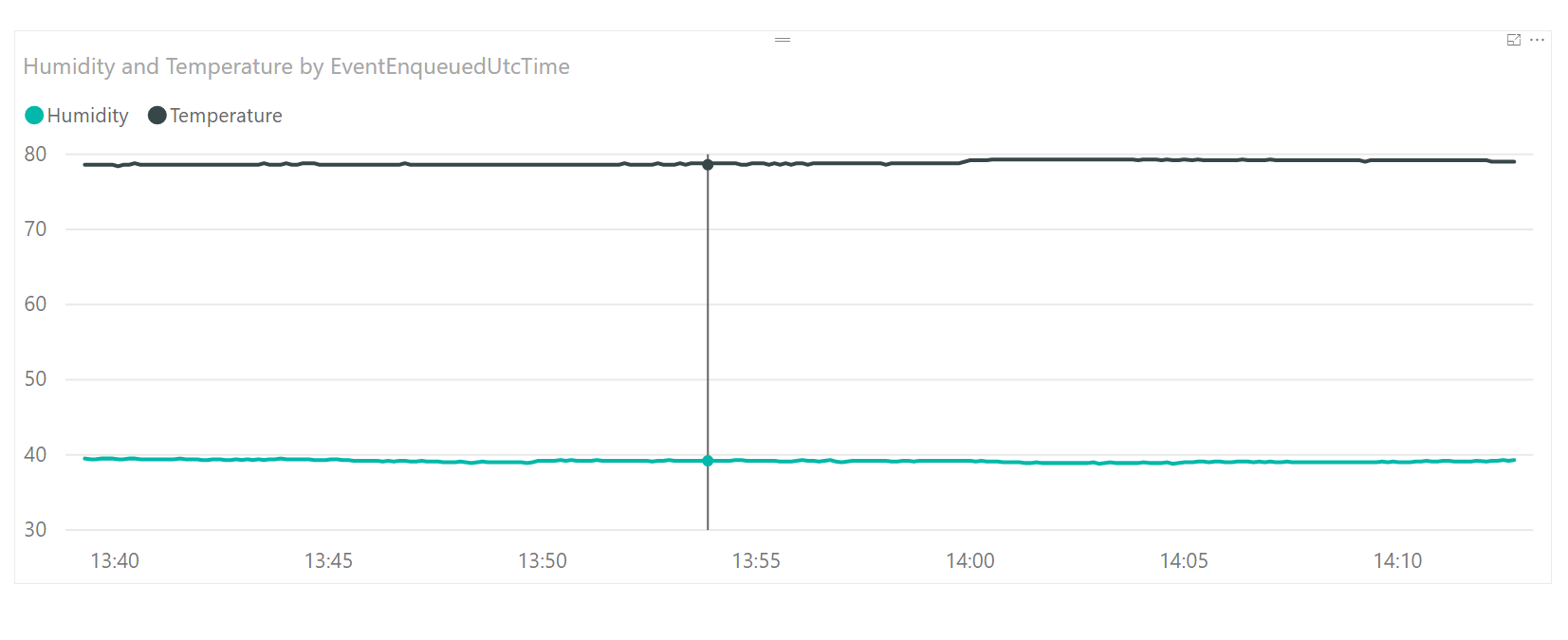


* In the Values box, drag Humidity and Temperature.
* Under Filters, drag DeviceID into the Visual level filters box and ensure that only DevKit is checked.

You should end up with a chart design that looks like this:



This will display a line chart like this:

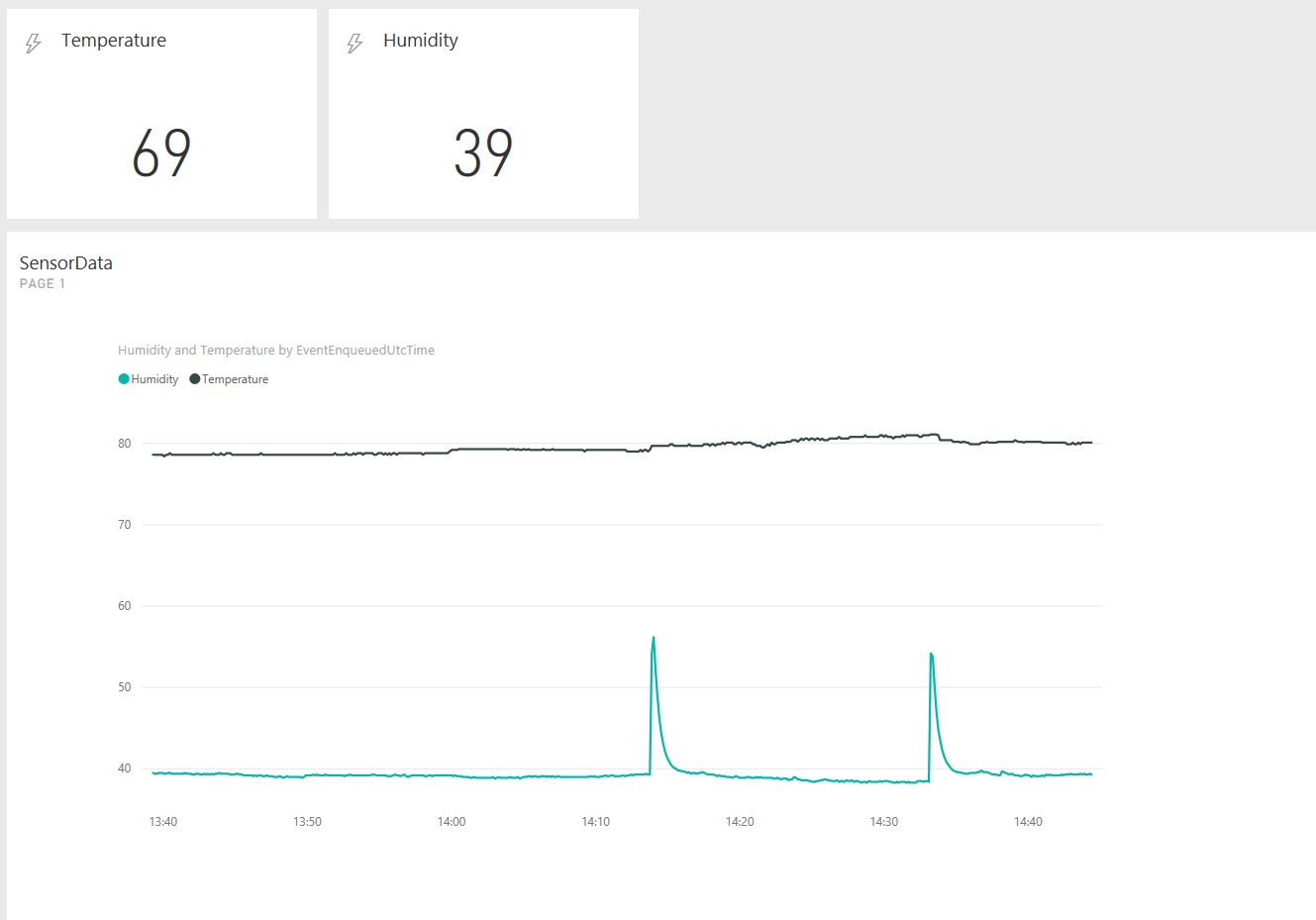


You can drag the chart around to make it larger.

In order to pull the latest data from SQL, click the Refresh button on the task bar. Breathe on the sensor a couple of times, then refresh the data. You will see the lines on the chart move around from the updates coming from SQL.

If you signed in to PowerBi Desktop with the same account as PowerBI.com, you can publish this dashboard to your online workspace and share it. Click the Publish button to do that and specify My Workspace.

Once you’ve published it in PowerBI.com, you can create a historical data report and pin it to the live streaming dashboard to get a combination of the two. Have a play around and see what you can create.



I hope you enjoyed the lab!