**Azure IoT Hub**

**(Azure Telemetry Service)**



**Submitted by**

Name of the Students University Roll No

Deepraj Rakshit 16500218048

Debojyoti Roy

Subhayu Kumar Bala

**Under the supervision of**

**Mr. Soumya Raha**

**Dept. of IT Computer Science**

**Ardent Computech Pvt. Ltd.**

**(Project Guide)**

Report submitted in partial fulfilment of the requirements for the

degree of Bachelor of Technology of Maulana Abul

Kalam Azad University of Technology.

Department of Bachelor of Technology

Calcutta Institute of Engineering and Management

24/1A, Chandi Ghosh Road, Tollygunge

Kolkata-700040

**Department of Bachelors of Technology**

**Calcutta Institute of Engineering and Management**

**24/1A, Chandi Ghosh Road, Tollygunge**

**Kolkata-700040**

**CERTIFICATE OF RECOMMENDATION**

We hereby recommend that the thesis prepared under the supervision of

Soumya Raha entitled as “**Azure IoT Hub**” can be accepted in

partial fulfilment of the requirements for the degree of

“**Bachelor of Technology**”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Nilanjan Dey Mr. Soumya Raha

Dean Dept. of IT Computer Science

C.I.E.M Ardent Computech Pvt. Ltd.

(Project Guide)

**Calcutta Institute of Engineering and Management**

**24/1A, Chandi Ghosh Road, Tollygunge**

**Kolkata-700040**

***Affiliated to***

**MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY**

**CERTIFICATE OF APPROVAL\***

**(Bachelor of Technology)**

This project report is hereby approved as a creditable study of Bachelor of Technology subject carried out and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree for which it has been submitted. It is to be understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed and conclusion drawn therein but approve the project report only for the purpose for which it has been submitted.

**COMMITTEE ON FINAL** 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXAMINATION FOR** 2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EVALUATION OF** 3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PROJECT REPORT** 4. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*Only in case report is approved

**ACKNOWLEDGEMENT**

We wish to extend our heartfelt gratitude and sincere thanks to Mr. Soumya Raha, Dept. of IT Computer Science, Ardent Computech Pvt. Ltd. For his constant support and encouragement given throughout the development of the project.

We thank Dr. Nilanjan Dey, Dean of Calcutta Institute of Engineering and Management, Dr. Amitabha Das, Principal of Calcutta Institute of Engineering and Management for their kind support and inspiration given.

Last but not least our sincere thanks to our parents, family members and friends for their continuous support, inspiration and encouragement without which this project would not have been successful.

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (DEEPRAJ RAKSHIT)
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (DEBOJYOTI ROY)
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(SUBHAYU KUMAR BALA)

**INDEX**

1. **Abstract**
2. **Introduction**

2.1] What is Azure IoT Hub?

2.2] Scaling solution

2.3] Securing communications

2.4] Route Device Data

2.5] Integrate with other devices

2.6] Configuring and controlling devices

2.7] Making solution highly available

2.8] Connecting Devices

2.9] Quotas and Limits

1. **Sending Telemetry from a device to an IoT Hub and reading it with a backend application**

3.1] Opening Azure Cloud Shell

3.2] Pre-Requisites

3.3] Creating an IoT Hub

3.4] Registering a Device

3.5] Sending simulated telemetry

3.6] Read the telemetry from your hub

1. **Conclusion**
2. **Bibliography**

**Abstract**

The purpose of the project entitled as “Azure IoT Hub” titled as “Azure Telemetry Service” is to focus on Microsoft Azure’s component, i.e., Azure IoT Hub. It’s one of the unique features of Azure Cloud which helps us to connect with other devices or microcontrollers (Raspberry) from a VMac or Virtual Machine. This whole project is based on IoT Hub Integration. In this Project we opened an Azure account and inside the portal we created an IoT Hub. Then we registered the external device which is to be connected and last send a telemetry message to the device from the machine to check if it is working or not. Python is also been used here for typing up the main source code which is needed for installing the device and checking up the messages send later. After all the resources are cleaned so that no further cost are being cut for that as Azure charges for every data sent and received.

**What is Azure IoT Hub?**

IoT Hub is a managed service, hosted in the cloud, that acts as a central message hub for bi-directional communication between your IoT application and the devices it manages. You can use Azure IoT Hub to build IoT solutions with reliable and secure communications between millions of IoT devices and a cloud-hosted solution backend. You can connect virtually any device to IoT Hub.

IoT Hub supports communications both from the device to the cloud and from the cloud to the device. IoT Hub supports multiple messaging patterns such as device-to-cloud telemetry, file upload from devices, and request-reply methods to control your devices from the cloud. IoT Hub monitoring helps you maintain the health of your solution by tracking events such as device creation, device failures, and device connections.

IoT Hub's capabilities help you build scalable, full-featured IoT solutions such as managing industrial equipment used in manufacturing, tracking valuable assets in healthcare, and monitoring office building usage.

**Scale your solution**

IoT Hub scales to millions of simultaneously connected devices and millions of events per second to support your IoT workloads. IoT Hub offers several tiers of service to best fit your scalability needs. Learn more by checking out the [pricing page](https://azure.microsoft.com/pricing/details/iot-hub/).

**Secure your communications**

IoT Hub gives you a secure communication channel for your devices to send data.

* Per-device authentication enables each device to connect securely to IoT Hub and for each device to be managed securely.
* You have complete control over device access and can control connections at the per-device level.
* The [IoT Hub Device Provisioning Service](https://docs.microsoft.com/azure/iot-dps/) automatically provisions devices to the right IoT hub when the device first boots up.
* Multiple authentication types support a variety of device capabilities:
  + SAS token-based authentication to quickly get started with your IoT solution.
  + Individual X.509 certificate authentication for secure, standards-based authentication.
  + X.509 CA authentication for simple, standards-based enrollment.

**Route device data**

Built-in message routing functionality gives you flexibility to set up automatic rules-based message fan-out:

* Use message routing to control where your hub sends device telemetry.
* There is no additional cost to route messages to multiple endpoints.
* No-code routing rules take the place of custom message dispatcher code.

**Integrate with other services**

You can integrate IoT Hub with other Azure services to build complete, end-to-end solutions. For example, use:

* [Azure Event Grid](https://docs.microsoft.com/azure/event-grid/) to enable your business to react quickly to critical events in a reliable, scalable, and secure manner.
* [Azure Logic Apps](https://docs.microsoft.com/azure/logic-apps/) to automate business processes.
* [Azure Machine Learning](https://docs.microsoft.com/azure/machine-learning/) to add machine learning and AI models to your solution.
* [Azure Stream Analytics](https://docs.microsoft.com/azure/stream-analytics/) to run real-time analytic computations on the data streaming from your devices.

**Configure and control your devices**

You can manage your devices connected to IoT Hub with an array of built-in functionality.

* Store, synchronize, and query device metadata and state information for all your devices.
* Set device state either per-device or based on common characteristics of devices.
* Automatically respond to a device-reported state change with message routing integration.

**Make your solution highly available**

There's a 99.9% [Service Level Agreement for IoT Hub](https://azure.microsoft.com/support/legal/sla/iot-hub/). The full [Azure SLA](https://azure.microsoft.com/support/legal/sla/) explains the guaranteed availability of Azure as a whole.

**Connect your devices**

Use the [Azure IoT device SDK](https://docs.microsoft.com/azure/iot-hub/iot-hub-devguide-sdks) libraries to build applications that run on your devices and interact with IoT Hub. Supported platforms include multiple Linux distributions, Windows, and real-time operating systems. Supported languages include:

* C
* C#
* Java
* Python
* Node.js.

IoT Hub and the device SDKs support the following protocols for connecting devices:

* HTTPS
* AMQP
* AMQP over WebSockets
* MQTT
* MQTT over WebSockets

If your solution cannot use the device libraries, devices can use the MQTT v3.1.1, HTTPS 1.1, or AMQP 1.0 protocols to connect natively to your hub.

If your solution cannot use one of the supported protocols, you can extend IoT Hub to support custom protocols:

* Use [Azure IoT Edge](https://docs.microsoft.com/azure/iot-edge/) to create a field gateway to perform protocol translation on the edge.
* Customize the [Azure IoT protocol gateway](https://github.com/Azure/azure-iot-protocol-gateway/blob/master/README.md) to perform protocol translation in the cloud.

**Quotas and limits**

Each Azure subscription has default quota limits in place to prevent service abuse, and these limits could impact the scope of your IoT solution. The current limit on a per-subscription basis is 50 IoT hubs per subscription. You can request quota increases by contacting support. For more details on quota limits:

* [Azure subscription service limits](https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits)
* [IoT Hub throttling and you](https://azure.microsoft.com/blog/iot-hub-throttling-and-you/)

# Sending telemetry from a device to an IoT hub and read it with a back-end application using Python

Python

IoT Hub is an Azure service that enables you to ingest high volumes of telemetry from your IoT devices into the cloud for storage or processing. In this quickstart, you send telemetry from a simulated device application, through IoT Hub, to a back-end application for processing.

The quickstart uses a pre-written Python application to send the telemetry and a CLI utility to read the telemetry from the hub. Before you run these two applications, you create an IoT hub and register a device with the hub.

## Open Azure Cloud Shell

Azure Cloud Shell is an interactive shell environment hosted in Azure and used through your browser. Azure Cloud Shell allows you to use either bash or PowerShell shells to run a variety of tools to work with Azure services. Azure Cloud Shell comes pre-installed with the commands to allow you to run the content of this article without having to install anything on your local environment.

To run any code contained in this article on Azure Cloud Shell, open a Cloud Shell session, use the **Copy** button on a code block to copy the code, and paste it into the Cloud Shell session with **Ctrl+Shift+V** on Windows and Linux, or **Cmd+Shift+V** on macOS. Pasted text is not automatically executed, so press **Enter** to run code.

You can launch Azure Cloud Shell with:

| **Option** | **Example/Link** |
| --- | --- |
| Select **Try It** in the upper-right corner of a code block. This **doesn't** automatically copy text to Cloud Shell. | Example of Try It for Azure Cloud Shell |
| Open [Azure Cloud Shell](https://shell.azure.com/) in your browser. |  |
| Select the **Cloud Shell** button on the menu in the upper-right corner of the [Azure portal](https://portal.azure.com/). | Cloud Shell button in the Azure portal |

## Prerequisites

The sample application you run in this quickstart is written using Python. Currently, the Microsoft Azure IoT SDKs for Python support only specific versions of Python for each platform. To learn more, see the [Python SDK Readme](https://github.com/Azure/azure-iot-sdk-python#important-installation-notes---dealing-with-importerror-issues).

This quickstart assumes you are using a Windows development machine. For Windows systems, only [Python 3.6.x](https://www.python.org/downloads/release/python-368/) is supported. The Python installer you choose should be based on the architecture of the system that you are working with. If your system CPU architecture is 32 bit, then download the x86 installer; for the 64-bit architecture, download the x86-64 installer. Additionally, make sure the [Microsoft Visual C++ Redistributable for Visual Studio 2019](https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads) is installed for your architecture (x86 or x64).

You can download Python for other platforms from [Python.org](https://www.python.org/downloads/).

You can verify the current version of Python on your development machine using one of the following commands:

python --version

python3 --version

Run the following command to add the Microsoft Azure IoT Extension for Azure CLI to your Cloud Shell instance. The IOT Extension adds IoT Hub, IoT Edge, and IoT Device Provisioning Service (DPS) specific commands to Azure CLI.

Azure CLI

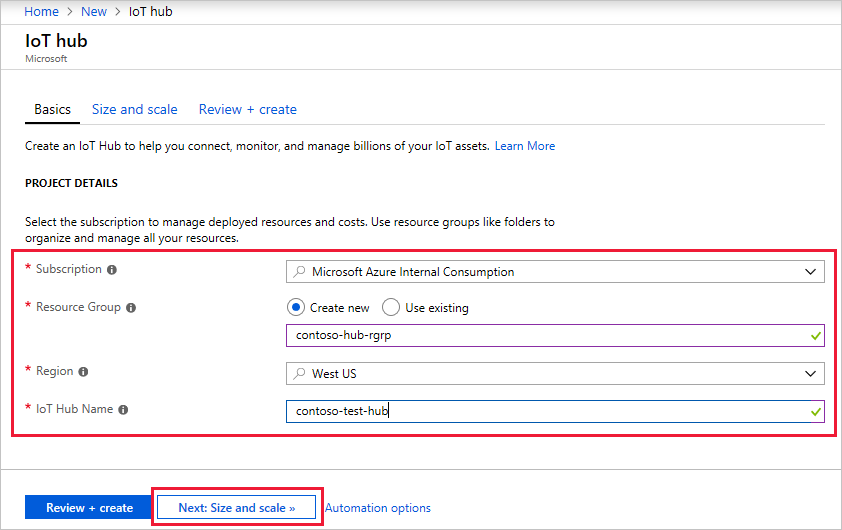
az extension add --name azure-cli-iot-ext

Download the sample Python project from <https://github.com/Azure-Samples/azure-iot-samples-python/archive/master.zip> and extract the ZIP archive.

## Create an IoT hub

This section describes how to create an IoT hub using the [Azure portal](https://portal.azure.com/).

1. Log in to the [Azure portal](https://portal.azure.com/).
2. Choose +**Create a resource**, then Search the Marketplace for the **IoT Hub**.
3. Select **IoT Hub** and click the **Create** button. You see the first screen for creating an IoT hub.



Fill in the fields.

**Subscription**: Select the subscription to use for your IoT hub.

**Resource Group**: You can create a new resource group or use an existing one. To create a new one, click **Create new** and fill in the name you want to use. To use an existing resource group, click **Use existing** and select the resource group from the dropdown list. For more information, see [Manage Azure Resource Manager resource groups](https://docs.microsoft.com/en-us/azure/azure-resource-manager/manage-resource-groups-portal).

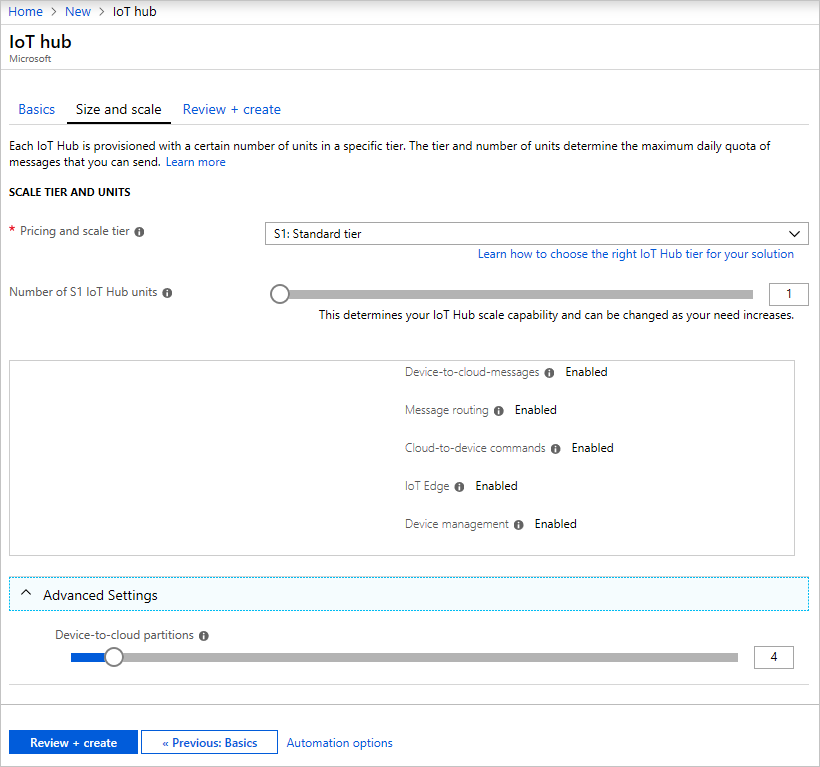
**Region**: This is the region in which you want your hub to be located. Select the location closest to you from the dropdown list.

**IoT Hub Name**: Put in the name for your IoT Hub. This name must be globally unique. If the name you enter is available, a green check mark appears.

**Important**

Because the IoT hub will be publicly discoverable as a DNS endpoint, be sure to avoid entering any sensitive or personally identifiable information when you name it.

1. Click **Next: Size and scale** to continue creating your IoT hub.



On this screen, you can take the defaults and just click **Review + create** at the bottom.

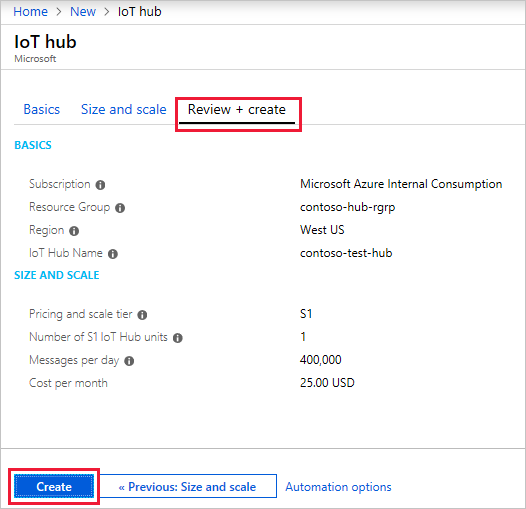
**Pricing and scale tier**: You can choose from several tiers depending on how many features you want and how many messages you send through your solution per day. The free tier is intended for testing and evaluation. It allows 500 devices to be connected to the IoT hub and up to 8,000 messages per day. Each Azure subscription can create one IoT Hub in the free tier.

**IoT Hub units**: The number of messages allowed per unit per day depends on your hub's pricing tier. For example, if you want the IoT hub to support ingress of 700,000 messages, you choose two S1 tier units.

For details about the other tier options, see [Choosing the right IoT Hub tier](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-scaling).

**Advanced / Device-to-cloud partitions**: This property relates the device-to-cloud messages to the number of simultaneous readers of the messages. Most IoT hubs only need four partitions.

1. Click **Review + create** to review your choices. You see something similar to this screen.



1. Click **Create** to create your new IoT hub. Creating the hub takes a few minutes.

## Register a device

A device must be registered with your IoT hub before it can connect. In this quickstart, you use the Azure Cloud Shell to register a simulated device.

1. Run the following command in Azure Cloud Shell to create the device identity.

**YourIoTHubName**: Replace this placeholder below with the name you choose for your IoT hub.

**MyPythonDevice**: This is the name given for the registered device. Use MyPythonDevice as shown. If you choose a different name for your device, you will also need to use that name throughout this article, and update the device name in the sample applications before you run them.

Azure CLI

az iot hub device-identity create --hub-name YourIoTHubName --device-id MyPythonDevice

1. Run the following commands in Azure Cloud Shell to get the device connection string for the device you registered:

**YourIoTHubName**: Replace this placeholder below with the name you choose for your IoT hub.

Azure CLI

az iot hub device-identity show-connection-string --hub-name YourIoTHubName --device-id MyPythonDevice --output table

Make a note of the device connection string, which looks like:

HostName={YourIoTHubName}.azure-devices.net;DeviceId=MyNodeDevice;SharedAccessKey={YourSharedAccessKey}

You use this value later in the quickstart.

## Send simulated telemetry

The simulated device application connects to a device-specific endpoint on your IoT hub and sends simulated temperature and humidity telemetry.

1. In a local terminal window, navigate to the root folder of the sample Python project. Then navigate to the **iot-hub\Quickstarts\simulated-device** folder.
2. Open the **SimulatedDevice.py** file in a text editor of your choice.

Replace the value of the CONNECTION\_STRING variable with the device connection string you made a note of previously. Then save your changes to **SimulatedDevice.py** file.

1. In the local terminal window, run the following commands to install the required libraries for the simulated device application:

cmd/sh

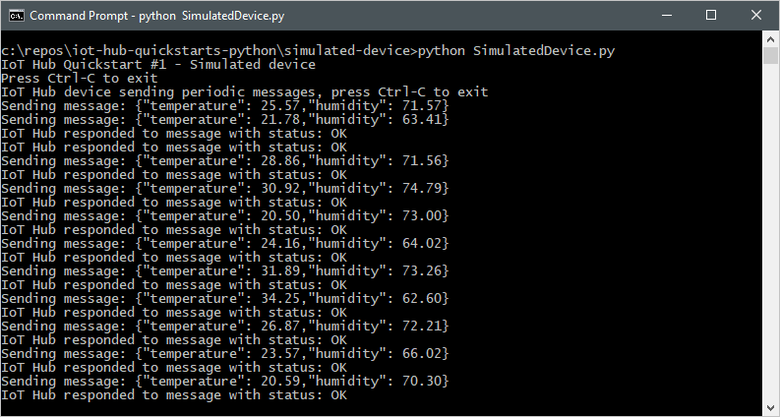
pip install azure-iothub-device-client

1. In the local terminal window, run the following commands to run the simulated device application:

cmd/sh

python SimulatedDevice.py

The following screenshot shows the output as the simulated device application sends telemetry to your IoT hub:



### To avoid the import iothub\_client error

The current version of the Azure IoT SDK for Python is a wrapper over [our C SDK](https://github.com/azure/azure-iot-sdk-c). It is generated using the [Boost](https://www.boost.org/) library. Because of that, it comes with several significant limitations. See more details [here](https://github.com/Azure/azure-iot-sdk-python#important-installation-notes---dealing-with-importerror-issues)

1. Check that you have the right version of [Python](https://github.com/Azure/azure-iot-sdk-python#important-installation-notes---dealing-with-importerror-issues). Be aware that only certain versions works fine for this sample.
2. Verify that you have installed the iothub client: pip install azure-iothub-device-client.

## Read the telemetry from your hub

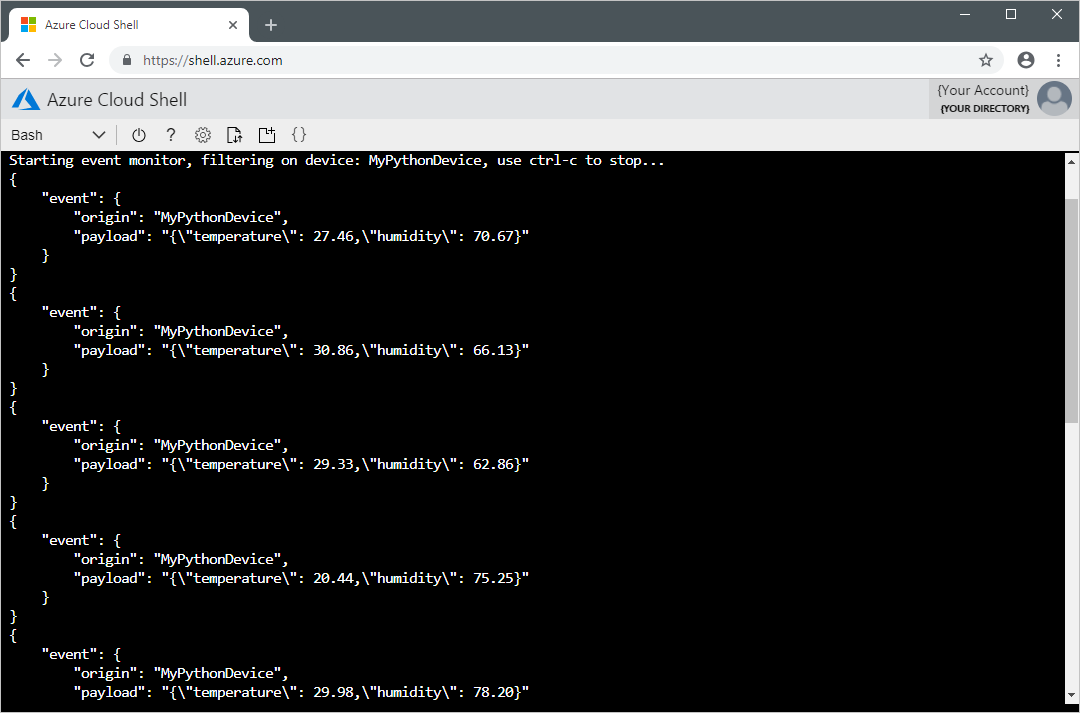
The IoT Hub CLI extension can connect to the service-side **Events** endpoint on your IoT Hub. The extension receives the device-to-cloud messages sent from your simulated device. An IoT Hub back-end application typically runs in the cloud to receive and process device-to-cloud messages.

Run the following commands in Azure Cloud Shell, replacing YourIoTHubName with the name of your IoT hub:

Azure CLI

az iot hub monitor-events --hub-name YourIoTHubName --device-id MyPythonDevice

The following screenshot shows the output as the extension receives telemetry sent by the simulated device to the hub:



## Clean up resources

If you will be continuing to the next recommended article, you can keep the resources you've already created and reuse them.

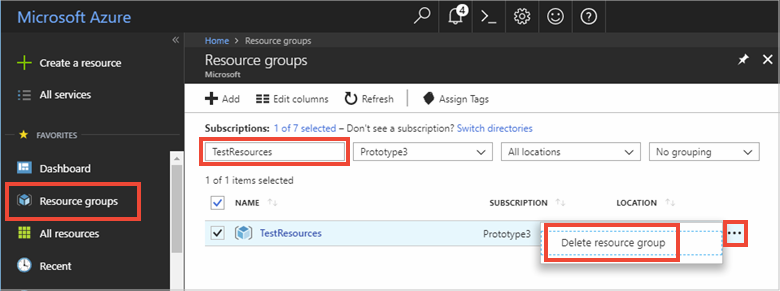
Otherwise, you can delete the Azure resources created in this article to avoid charges.

**Important**

Deleting a resource group is irreversible. The resource group and all the resources contained in it are permanently deleted. Make sure that you do not accidentally delete the wrong resource group or resources. If you created the IoT Hub inside an existing resource group that contains resources you want to keep, only delete the IoT Hub resource itself instead of deleting the resource group.

To delete a resource group by name:

1. Sign in to the [Azure portal](https://portal.azure.com/) and click **Resource groups**.
2. In the **Filter by name...** textbox, type the name of the resource group containing your IoT Hub.
3. To the right of your resource group in the result list, click **...** then **Delete resource group**.



1. You will be asked to confirm the deletion of the resource group. Type the name of your resource group again to confirm, and then click **Delete**. After a few moments, the resource group and all of its contained resources are deleted.

**CONCLUSION**

This project creates a Microsoft Azure account, with help of which we able to use Azure’s feature. It also focusses on creating IoT Hub. Then we have to connect the external device and send a telemetry message. The IoT Hub CLI extension can connect to the service-side **Events** endpoint on your IoT Hub. The extension receives the device-to-cloud messages sent from your simulated device. An IoT Hub back-end application typically runs in the cloud to receive and process device-to-cloud messages. We can delete the Azure resources created so to avoid extra charges.

**BIBLIOGRAPHY**

We would like to thank the authors and publisher of all the books which we have read to understand the topic and complete this project. Some of the mentioned books were very handy.

* 1. “Introducing Windows Azure for IT Professionals” by Mitch Tulloch
  2. “Deployment of Microsoft Azure Cloud Solutions” by Florian Klaffenbach
  3. “Azure IoT Development Cookbook” by Yatish Patil.

Along with the books, websites were also surfed for relevant information which was need for completion of this project.

1. https://www.youtube.com
2. https://azure.microsoft.com
3. https://www.edx.org
4. https://www.azureiotsolutions.com