



Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

[Read the guide](#)

[mpram / IoT_Academy](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#)

master ▾

...

[IoT_Academy / Month_1 / Day_1 / Azure IoT Foundation.md](#)



mpram Foundation

History

1 contributor

Raw

Blame



662 lines (397 sloc) | 24.9 KB

Internet of Things

1. Azure IoT Foundation, theory
2. Getting familiar with Azure Portal <https://ms.portal.azure.com/>

Content:

- Exercise 1: IoT Hub provisioning
 - Task 1: Provision IoT Hub through the Portal
 - Task 2: Provision IoT Hub through CLI
 - Task 3: Provision IoT Hub through VS Code
- Exercise 2: Devices
 - Task 1: Setting up a Device
 - Task 2: Setting up an IoT Edge Device
- Exercise 3: Deploying Modules
 - Task 1: Temperature Simulated Module
- Exercise 4: Telemetry Data
 - Task 2: Setting up Services
 - Task 2: Validating connectivity
 - Task 3: Connecting the services
- Exercise 5: Monitoring Remote Devices
 - Task 1: Setting up alerts
 - Task 2: Creating the Workflow
 - Task 3: Subscribe to the Events
- Exercise 6: Interacting with remote devices
 - Task 1: Invoke a direct method
 - Task 2: Invoke a direct method, checking the logs
- Exercise 7: Clean up
 - Task 1: Delete resources

Exercise 1: IoT Hub provisioning

Task 1: Provision IoT Hub through the Portal

During this exercise you will use 3 different tools to create three different IoT Hubs, after this exercise we will delete two and continue the rest of the workshop with the first IoT Hub created through the Portal.

1. In your browser, navigate to the [Azure portal](#), select **+Create a resource** in the navigation pane, enter `iot` into the **Search the Marketplace** box.
2. Select **IoT Hub** from the results, and then select **Create**.

The screenshot shows the Microsoft Azure 'New' blade. On the left, there's a sidebar with a red box around the '+ Create a resource' button. The main area has a search bar at the top with 'IoT Hub' typed into it, also highlighted with a red box. Below the search bar, 'IoT Hub' is listed in the search results. To the right of the search results, there are several quick links: 'Get started' (Windows Server 2016 Datacenter), 'Recently created' (Ubuntu Server 18.04 LTS), and 'AI + Machine Learning'.

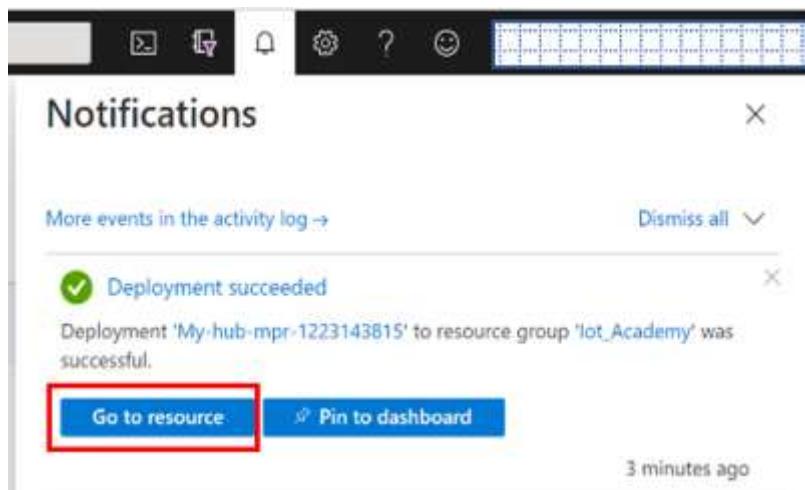
3. On the **IoT Hub** blade **Basics** tab, enter the following:

- **Subscription:** Select the subscription you are using for this hands-on lab.
- **Resource group:** Choose **Use existing** and select the resource group.
- **Region:** Select the location you are using for this hands-on lab.
- **IoT Hub Name:** Enter a unique name, such as `my-hub-SUFFIX`.

The screenshot shows the 'Basics' tab of the IoT Hub creation blade. It includes fields for Project details (Subscription dropdown, Resource group dropdown set to 'IoT_Academy', Region dropdown set to 'East US', and IoT hub name input field containing 'My-hub-mp'). At the bottom, there are navigation buttons: 'Review + create' (highlighted in blue), '< Previous', 'Next: Networking >', and 'Automation options'.

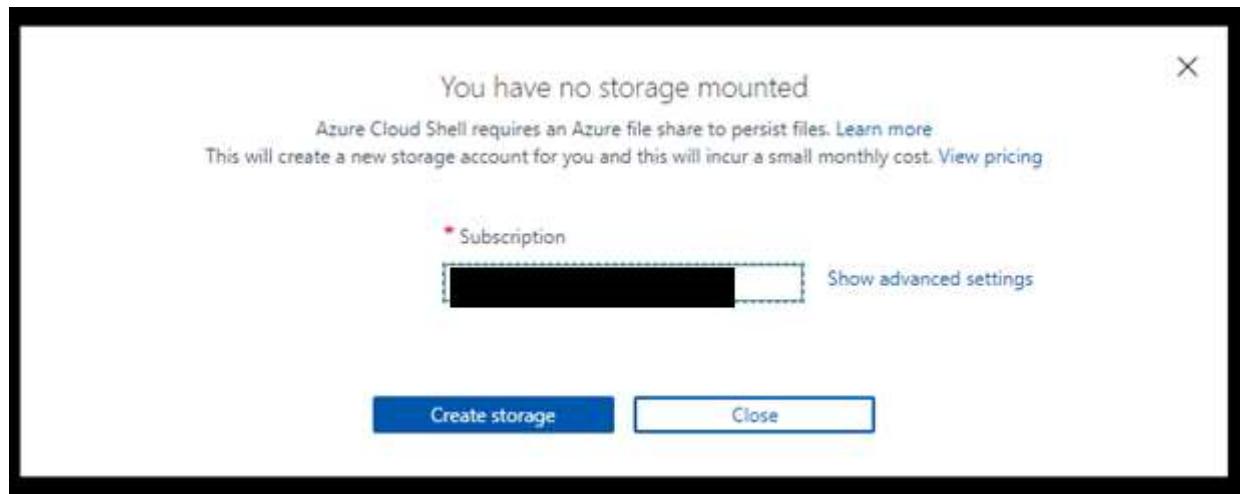
- Select **Next: Size and Scale**.
- On the **Size and scale** tab, accept the default Pricing and scale tier of **S1: Standard tier**, and select **Review + create**.

- Select **Create** on the **Review + create** blade.
4. When the IoT Hub deployment is completed, you will receive a notification in the Azure portal. Select **Go to resource** in the notification.



Task 2: Provision IoT Hub through CLI

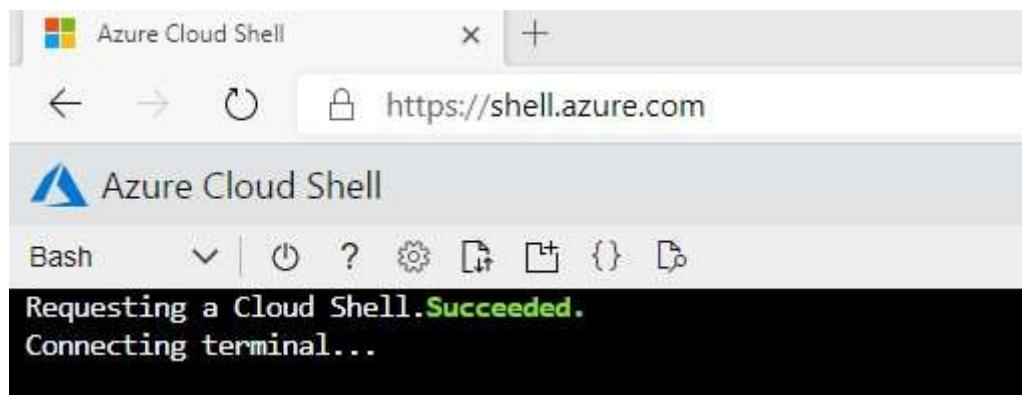
If you never used before you will be prompted to mount an storage account, click **Create Storage** to continue. If you used before, you will skip this step.



Open cloud with the below link

<https://shell.azure.com/>

Change to **Bash** access



Once you are login run the following command to create an IoT Hub.

```
az iot hub create --name {your iot hub name} \
--resource-group {your resource group name} --sku S1
```

Verify your IoT Hub has been created in the Portal.

To delete the IoT Hub just created you can use a delete command:

```
az iot hub delete --name {your iot hub name} --resource-group {your resource group}
```

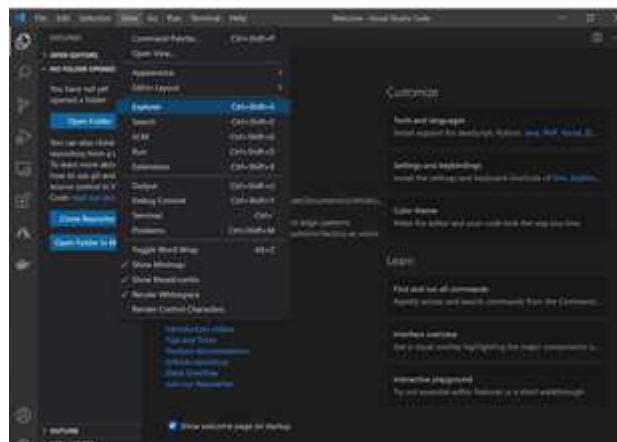
Task 3: Provision IoT Hub through VS Code

Last we will use another tool to also create an IoT Hub, in this case VS Code. For this task make sure you download VS Code in advance. Download link:

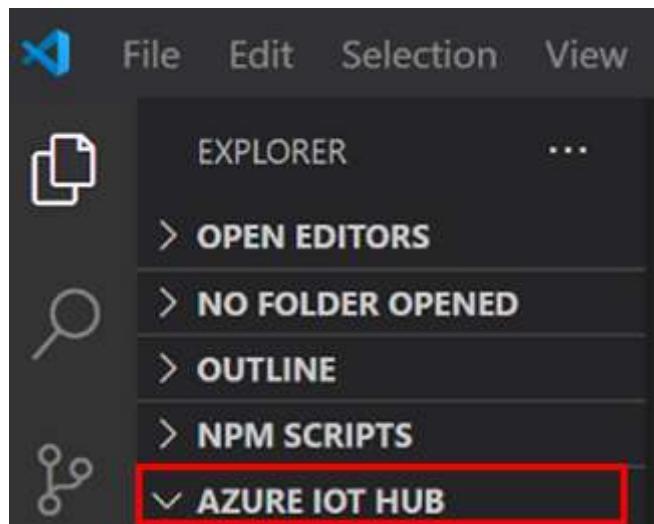
<https://code.visualstudio.com/Download>

1. Install IoT Tools extension for VS Code: <https://marketplace.visualstudio.com/items?itemName=vsciot-vscode.azure-iot-tools>

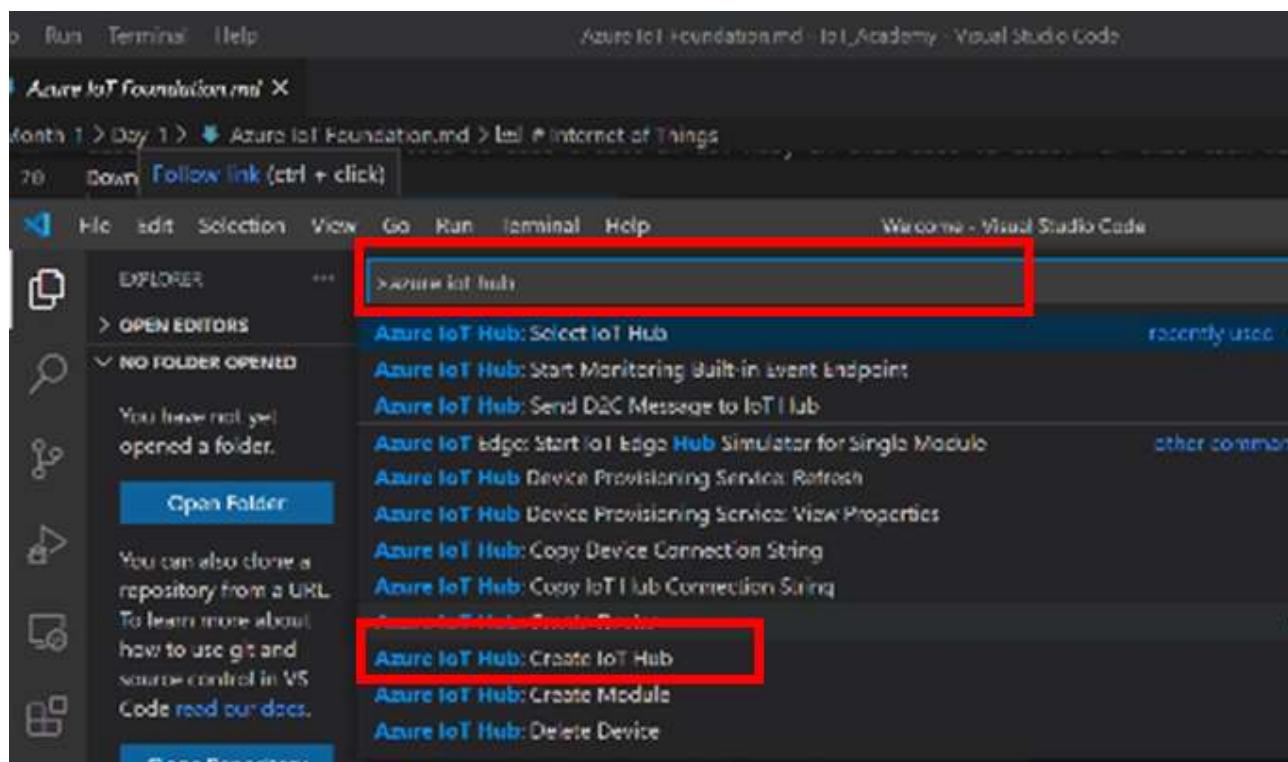
2. Go To View Explorer



Now you should be able to see the Azure IoT Hub



To create a new IoT Hub Go to the menu **View** on the top toolbar then select **Command Palette**



Type **Azure IoT Hub** in the new window, then you will see the list of commands available select **Azure IoT Hub: Create IoT Hub** and click Enter. Then you will need to select the following parameters:

- **Subscription:** Select the subscription you are using for this hands-on lab.
- **Resource group:** Use existing and select the resource group.
- **Location:** Select the location you are using for resources in this hands-on lab.

- **SKU:** Select S1.
- **Name:** Assign a name to the IoT Hub `vscodeiotcreate` add suffix as needed.

After the creation process you should be able to see the new IoT Hub in Azure Portal and in VS Code.

Exercise 2: Devices

During this exercise you will learn how to set up and edge device and connect your device to IoT Hub to start streaming data.

Task 1: Setting up a Device

From Azure Portal select the IoT Hub created through VS Code previously, scroll down to **Automatic Device Management** then select **Add an IoT Edge device**

The screenshot shows the Azure portal interface for managing an IoT Edge device. The top navigation bar includes 'Microsoft Azure (Preview)', a search bar, and various icons. The main title is 'vscodeiothub | IoT Edge'. On the left, there's a sidebar with sections like 'Explorers', 'IoT devices', 'Automatic Device Management' (which is highlighted with a red box), and 'Messaging'. The main content area shows an 'IoT Edge devices' section with a table header for 'Field', 'Operator', and 'Value'. At the bottom of this section is a 'Query devices' button. Above the table, there's a note about deploying Azure services to on-premises devices. The top navigation bar also has buttons for 'Create Deployment', 'Create Layered Deployment', 'Refresh', and 'Delete'.

In the new window select a name for your device **my-device-SUFFIX** and click **Save**

Create a device

|  Directory: Microsoft



Find Certified for Azure IoT devices in the Device Catalog



Device ID * 

my-device



Authentication type 

Symmetric key X.509 Self-Signed

Primary key 

Enter your primary key

Secondary key 

Enter your secondary key

Auto-generate keys 



Connect this device to an IoT hub 

Enable Disable

Parent device 

No parent device

[Set a parent device](#)

Child devices 

0

[Choose child devices](#)

Save

After Creation your device will be available with new information, click on the device

The screenshot shows the Azure IoT Edge devices management interface. On the left, there's a sidebar with various navigation options like IoT Hub, Query explorer, IoT devices, etc. The main area is titled 'IoT Edge devices' and shows a table of devices. One device, 'my-device', is highlighted with a red box around its row. The table columns include Device ID, Runtime Response, IoT Edge Module Count, Connected Client Count, and Deployment Count.

Now, copy and paste in the a notepad the connection string of your device, you will need this to connect your device to IoT hub, we will use this connection string in the next task.

This screenshot shows the detailed configuration page for the device 'my-device'. It includes fields for Device ID, Primary Key, Secondary Key, and Primary Connection String. The Primary Connection String field is highlighted with a red box. Below the device details, there are tabs for Modules, IoT Edge hub connections, and Deployments and Configurations. The Modules table lists two modules: \$edgeAgent and \$edgeHub.

Task 2: Setting up an IoT Edge Device

In this exercise we will set up an IoT Edge device, for that we will create VM using Ubuntu Server acting as one.

1. From Azure Portal select **Create resource** then from the most Popular list select **Ubuntu Server 18.04 LTS** if you don't see it type the same in the Search window.

The screenshot shows the Microsoft Azure (Preview) portal's 'New' blade. At the top, there are navigation links for 'Home', 'New', and a search bar. Below this, there is a 'Search the Marketplace' input field. Under 'Azure Marketplace', there are two tabs: 'See all' and 'Popular'. A sidebar on the left lists categories: 'Get started', 'Recently created', 'AI + Machine Learning', 'Analytics', 'Blockchain', 'Compute', 'Containers', 'Databases', 'Developer Tools', and 'More'. To the right, there are several service cards: 'Windows Server 2016 Datacenter' (Quickstarts + tutorials), 'Ubuntu Server 18.04 LTS' (Learn more, highlighted with a red box), 'Web App' (Quickstarts + tutorials), 'SQL Database' (Quickstarts + tutorials), and 'Function App'. Each card has a small icon representing the service.

2. Then you will need to complete the following parameters in the **Basics** tab:

- **Subscription:** Select the subscription you are using for this hands-on lab.
- **Resource group:** Use existing and select the resource group.
- **Virtual Machine Name:** edgedevice+SUFFIX
- **Region:** Select the location you are using for resources in this hands-on lab.
- **Availability Options:** Select **No Infrastructure redundancy required**.
- **Image:** Default it is ok.
- **Size:** Keep the default selection
- **Authentication Type:** Select **Password**
- **Username:** iotacademy
- **Password:** MSFTacademy01!
- **Allow selected ports:** SSH 22

In **Management** Tab disable monitoring for this lab.

Last select **Review + Create** after successfull validation you should be able to click **Create**

3. Once the resource is available click in the Virtual Machine, you should see in the Overview section the Public IP to connect, copy the IP.

vmacademymp

Virtual machine | Directory: Microsoft

Search (Ctrl+ /)

Connect Start Restart Stop Capture Delete Refresh Open in mobile

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Tags (change) Click here to add tags

Resource group (change) : Demo

Status : Running

Location : Central US

Subscription (change) : [REDACTED]

Subscription ID : [REDACTED]

Operating system : Linux (ubuntu 18.04)

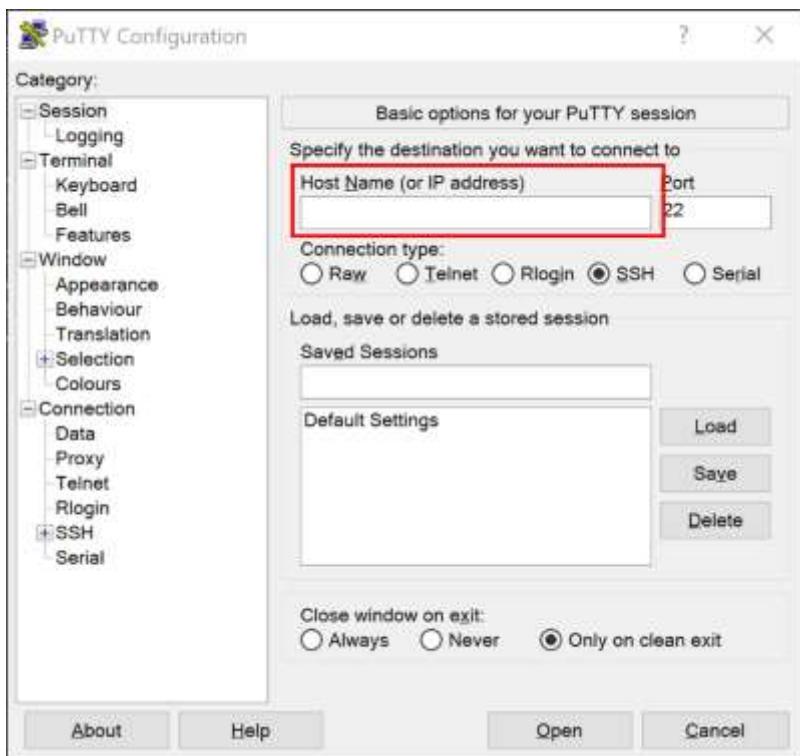
Size : Standard D2s v3 (2 vcpus, 8 GiB memory)

Public IP address : [REDACTED]

Virtual network/subnet : Demo-vnet/default

DNS name : Configure

4. In your laptop open Putty, locally, to connect to the Virtual Machine just created and configure IoT Edge. Paste the IP in the following window:



Note: If your laptop doesn't have putty installed, you can download it from here:
<https://www.putty.org/>

5. Once connected you will ask to enter user and password. After successful login, let's start to install the Edge Runtime

Install the repository configuration that matches your device operating system.

```
curl https://packages.microsoft.com/config/ubuntu/18.04/multiarch/prod.list > ./microsoft-iot-edge-repo.list
```

Copy the generated list to the sources.list.d directory.

```
sudo cp ./microsoft-prod.list /etc/apt/sources.list.d/
```

Install the Microsoft GPG public key.

```
curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg  
sudo cp ./microsoft.gpg /etc/apt/trusted.gpg.d/
```

Azure IoT Edge software packages are subject to the license terms located in each package (usr/share/doc/{package-name} or the LICENSE directory). Read the license terms prior to using a package. Your installation and use of a package constitutes your acceptance of these terms. If you do not agree with the license terms, do not use that package.

Install a Container Engine Update package lists on your device.

```
sudo apt-get update
```

Install the Moby engine.

```
sudo apt-get install moby-engine
```

If you want to install the most recent version of the security daemon, use the following command that also installs the latest version of the libiothsm-std package:

```
sudo apt-get install iotedge
```

Configure the connection to your IoT Hub, we will apply the connection string you copied in Task 1. Open the configuration file in your device to edit the connection string with the following command.

```
sudo nano /etc/iotedge/config.yaml
```

Once in the nano editor, scroll down to **Manual Provisioning configuration using a connection string** then replace the `device_connection-string` variable with the connection string from the task 1.

```

iotacademympc@iotacademympc ~
GNU nano 2.9.3 /etc/iotedge

# The value should be specified as a URI.
# Ex. when specifying a PEM encoded certificate file, the URI
# should be specified as file:///path/identity_certificate.pem
# identity_pk - Optional. The Edge device identity private key
# entry should only be specified when an Edge device
# is configured for X.509 authentication.
# The value should be specified as a URI.
# Ex. when specifying a PEM encoded private key file, the URI
# should be specified as file:///path/identity_key.pem

# External Settings
# endpoint - Required. Value of the endpoint used to retrieve device specific
# information such as its IoT hub connection information.

# Dynamic Re-provisioning Settings
# dynamic_reprovisioning - Optional. A flag to opt-in to the dynamic re-provisioning
# feature. If enabled, IoT Edge on detecting a possible
# device re-provisioning event will shut down the daemon.
# This is so that on the next daemon startup, the device is
# set up with the new provisioning information of the device in
# in IoT Hub.
# For the external provisioning mode specifically, the daemon
# will notify the external provisioning endpoint about the
# re-provisioning event before shutting down.

# Manual provisioning configuration using a connection string
# provisioning:
#   source: "manual"
#   device_connection_string: "REDACTED" REDACTED
#   dynamic_reprovisioning: false

# Manual provisioning configuration using an X.509 identity certificate
# provisioning:
#   source: "manual"
#   authentication:
#     method: "x509"
#     iothub_hostname: "<REQUIRED IOTHUB HOSTNAME>"
#     device_id: "<REQUIRED DEVICE ID PROVISIONED IN IOTHUB>"
#     identity_cert: "<REQUIRED URI TO DEVICE IDENTITY CERTIFICATE>"
#     identity_pk: "<REQUIRED URI TO DEVICE IDENTITY PRIVATE KEY>"
#   dynamic_reprovisioning: false

# DPS TPM provisioning configuration
# provisioning:
#   source: "dps"
#   global_endpoint: "https://global.azure-devices-provisioning.net"
#   scope_id: "<SCOPE_ID>"
#   attestation:

```

After configuring your connectivity, press **CrtL+X** to close the file and select **Y** to save the changes

Now restart your edge daemon

```
sudo systemctl restart iotedge
```

In a few minutes you should receive a **Running** Status after executing the following command:

```
sudo iotedge list
```

Exercise 3: Deploying Modules

Task 1: Temperature Simulated Module

In Azure Portal, click on your IoT Hub created in previous steps, under **Automatic Device Provisioning** select your Edge Device, then select **Set Modules**

The screenshot shows the 'Set modules' section of the Azure IoT Hub device configuration page. It includes fields for Device ID (my-device-mpr), Primary Key, and Secondary Key, all of which are currently empty.

Then to add the temperature simulator, select **Add** then you will see multiple options, select **+ Marketplace Module**

The search box will appear, type **Simulated Temperature Sensor** now the Microsoft IoT Edge Module will be available to select, click on it and now will be available for configuration and deployment.

This screenshot shows the 'Set modules on device: my-device-mpr' page. The 'Add' button is highlighted with a red box. Below it, the 'SimulatedTemperatureSensor' module is listed in the list of available modules.

Let's analyze the Routes(deck), then **Review + create**

In a few minutes you will see 3 modules running in your Edge Device, you have two ways of verifying

- Check the modules running in your Edge Device:

Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
\$edgeAgent	IoT Edge System Module	✓ Yes	⊖ No	--	--
\$edgeHub	IoT Edge System Module	✓ Yes	⊖ No	--	--
SimulatedTemperatureSensor	IoT Edge Custom Module	✓ Yes	⊖ No	--	--

- Other way to visualize the modules is, go back to your Ubuntu VM, Open Putty and run again the following command

```
sudo iotedge list
```

Now you should see all the modules up and running in your Ubuntu Machine as shown below:

```
iotacademypr@vmacademypr:~$ sudo iotedge list
NAME          STATUS    DESCRIPTION      CONFIG
SimulatedTemperatureSensor  running   Up 1 second  mcr.microsoft.com/azureiotedge-simulated-temperature-sensor:1.0
edgeAgent      running   Up 9 seconds  mcr.microsoft.com/azureiotedge-agent:1.0
edgeHub        running   Up 3 seconds  mcr.microsoft.com/azureiotedge-hub:1.0
iotacademypr@vmacademypr:~$
```

Exercise 4: Telemetry Data

Task 1: Validating connectivity

First step it will be to validate IoT Hub is receiving data, open cloud shell, clicking in the following icon, top right in Azure Portal:



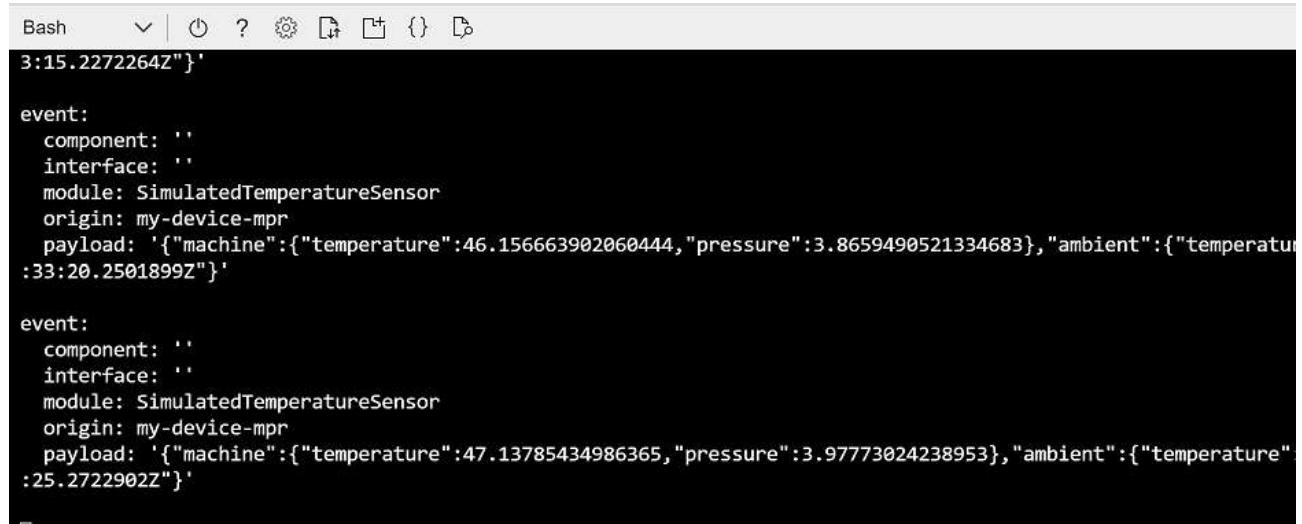
Run the az extension add command to add the Microsoft Azure IoT Extension for Azure CLI to your CLI shell. The IOT Extension adds IoT Hub, IoT Edge, and IoT Device Provisioning Service (DPS) specific commands to Azure CLI.

```
az extension add --name azure-iot
```

After you install the Azure IOT extension, you don't need to install it again in any Cloud Shell session.

```
az iot hub monitor-events --hub-name {YourIoTHubName} --output table
```

After running the above command you should be able to see telemetry data sent from our device to IoT Hub as shown below:



A screenshot of a terminal window titled "Bash". The window displays two lines of JSON-formatted telemetry data. Each line represents an "event" with fields: component, interface, module, origin, and payload. The payload contains machine and ambient temperature and pressure data. The timestamp in the first line is "3:15.2272264Z".

```
Bash
3:15.2272264Z"}'

event:
  component: ''
  interface: ''
  module: SimulatedTemperatureSensor
  origin: my-device-mpr
  payload: '{"machine":{"temperature":46.156663902060444,"pressure":3.8659490521334683},"ambient":{"temperature":33.20.2501899Z"}'

event:
  component: ''
  interface: ''
  module: SimulatedTemperatureSensor
  origin: my-device-mpr
  payload: '{"machine":{"temperature":47.13785434986365,"pressure":3.97773024238953},"ambient":{"temperature":25.2722902Z"}'
```

Task 2: Setting up Services

Create an Stream Analytics Job and the Storage Account After you validated IoT Hub is receiving data, next steps will be to create the services needed to analyze data further.

Click in **Create** button in Azure Portal, in the search box type **Stream Analytics Job**, then **Create** again.

Fill the fields in the form:

New Stream Analytics job

This will create a new Stream Analytics job. You will be charged according to your usage.

Job name *

Subscription *

Resource group *

 [Create new](#)

Location *

 Central US

Hosting environment

Cloud Edge

Streaming units (1 to 192) 3

Secure all private data assets needed by this job in my Storage account.

- **Job name:** satrainingSUFFIX
- **Subscription:** Select The subscription you are using for this training
- **Resource Group:** Select the Resource Group you are using for this training.
- **Location:** Select the Location you are using for this training.
- **Hosting Environment:** Cloud
- **Streaming Units:** Default.

Then click **Create**

Next step you will create an Storage Account to use as an output for your data.

Again, click in **Create** button in Azure Portal, in the search box type **Storage Account**, then **Create** again.

Fill the fields in the form:

Create storage account

Basics Networking Data protection Advanced Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below.

[Learn more about Azure storage accounts](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *



Resource group *



[Create new](#)

Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name * ⓘ

Location *



(US) Central US

Performance ⓘ

Standard Premium

Account kind ⓘ



StorageV2 (general purpose v2)

Replication ⓘ



Locally-redundant storage (LRS)

[Review + create](#)

< Previous

Next : Networking >

Fill the fields in the form:

- **Subscription:** Select The subscription you are using for this training
- **Resource Group:** Select the Resource Group you are using for this training.
- **Storage account name:** sadataoutputSUFFIX
- **Location:** Select the Location you are using for this training.
- **Performance:** Default
- **Account kind:** Default.
- **Replication:** Locally Redundant Storage(LRS)

Then click **Review and Create** after succesfull validation, click on **Create**.

Task 3: Connecting the services

Connecting the services. From your Resource Group select the Stream Analytics Job created above. In the section **Job Topology** Select + Add Stream input, IoT Hub.

The screenshot shows the Azure Stream Analytics Job Topology interface. On the left, under 'Job topology', the 'Inputs' tab is selected (marked with a red circle 1). A new input is being added, indicated by the 'Add stream input' button (marked with a red circle 2). The right side of the screen displays the 'IoT Hub' configuration dialog (marked with a red circle 3). The dialog fields include:

- Name:** (Input alias) [Empty]
- Source type:** (New input)
- Subscription:** [Subscription dropdown]
- IoT Hub:** [IoT Hub dropdown]
- Consumer group:** [\$Default]
- Shared access policy name:** iothubowner
- Shared access policy key:** [Redacted]
- Endpoint:** Messaging
- Partition key:** [Empty]
- Event serialization format:** JSON

A note at the bottom right of the dialog states: 'The selected resource and the stream analytics job are located in different regions. You will be billed to move data between regions.'

- **Input Alias:** inpuuthub
- Make sure **Select IoT Hub from your subscription** it is selected.
- **Subscription:** Select The subscription you are using for this training
- **IoT Hub:** Select the IoT Hub created in Exercise #1, Task #1, **My-hub-SUFFIX**
- **Consumer Group:** \$Default
- **Shared access policy name:** iothubowner Keep the rest of the fields as default and click **Save**

Once the Input is created, we will need an output. Back to **Job Topology**, select **Outputs** then + Add. From the menu select **Blob Storage/ADLS Gen2**

Blob storage/ADLS Gen2

X

New output

Output alias *

- Provide Blob storage/ADLS Gen2 settings manually
 Select Blob storage/ADLS Gen2 from your subscriptions

Subscription

Storage account *

Container *

- Create new Use existing

Authentication mode

Path pattern

Date format

Time format

Save

Cancel

- **Output alias:** storageaccountdata
- Make sure **Select Blob storage/ADLS Gen2 from your subscriptions** it is selected
- **Subscription:** Select The subscription you are using for this training
- **Storage account:** Select the storage account created in previous step.
- **Container:** Select **Create new** assign name as **data**
- **Authentication Mode:** Connection String

Keep the remaining options as default and click **Save**

Next step should be to define the query, for this go back to **Job Topology** click on **Query** a new window will open with the Input and Output defined in previous steps:

Copy the following command in the window"

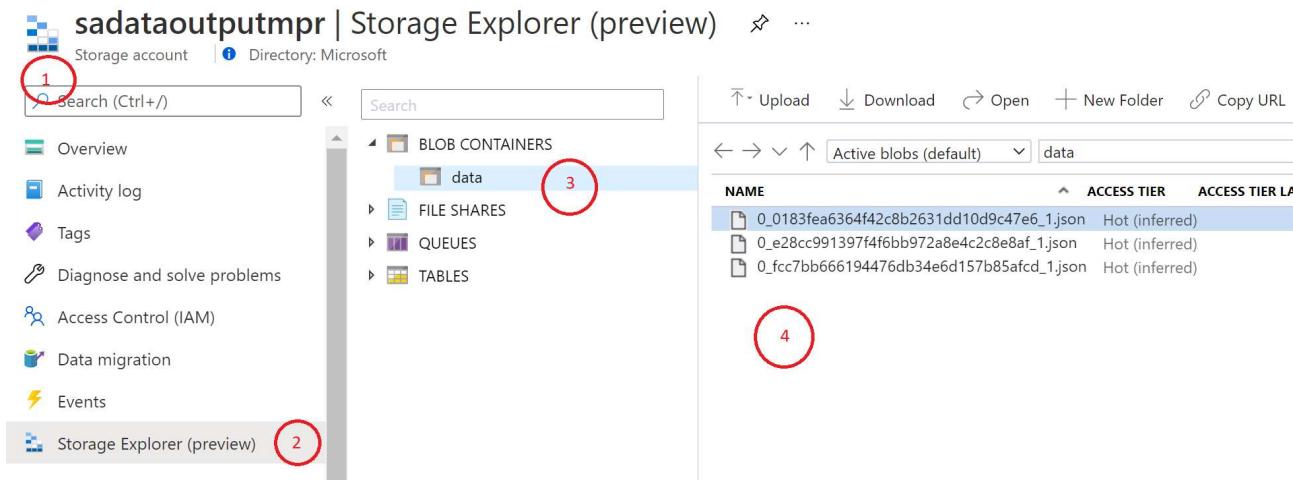
```
SELECT
    *
INTO
    storageaccountdata
FROM
    inpuhub
```

Click on **Save** query, in the **Overview** click **Start** to trigger the job. Select **Now** and then **Start** again.



In a few minutes your Storage account should be receiving data in the container just created.

1. Go to your Storage Account
2. Click Storage Explorer(Preview)
3. Click on the container just created.
4. You should see the data already available in json format, download to see it.



Exercise 5: Monitoring Remote Devices

Task 1: Setting up alerts

For this exercise will need a Logic App to orchestrate the process, click in **Create** button in Azure Portal, in the search box type **Logic App**, then **Create**.

Fill in the fields in the form:

Create workflows leveraging hundreds of connectors and the visual designer. [Learn more](#) 

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *



Resource group *



[Create new](#)

Instance details

Logic app name *



Region *



Associate with integration service environment 



Integration service environment



Enable log analytics 



Log Analytics workspace



[Review + create](#)

< Previous : Basics

Next : Tags >

[Download a template for automation](#) 

Fill the fields in the form:

- **Subscription:** Select The subscription you are using for this training
- **Resource Group:** Select the Resource Group you are using for this training.
- **Logic app name:** devicesalertSUFFIX
- **Region:** Select the same region you are using during this training.

Keep the remaining selections as default. Then select **Review + Create**, after succesfull validation click **Create**

In the Logic Apps Designer, page down to see Templates. Choose **Blank Logic App** so that you can build your logic app from scratch.

A trigger is a specific event that starts your logic app. For this tutorial, the trigger that sets off the workflow is receiving a request over HTTP.

In the connectors and triggers search bar, type **HTTP**.

Scroll through the results and select **Request - When a HTTP request is received**.

The screenshot shows the Microsoft Logic Apps connector search interface. At the top, there is a search bar with the text "http" and a magnifying glass icon. Below the search bar, there are tabs: "For You", "All" (which is selected), "Built-in", "Standard", "Enterprise", and "Custom". The main area displays a grid of connectors:

Request	Office 365 Outlook	SharePoint	FTP	OneDrive	Aquaforest PDF	Azure Blob Storage
Azure Data Explorer	Azure DevOps	boomapp connect	Cloudmersive Data...	Content Moderator	CRM Bot	emfluence Marketing...

Below the connector grid, there is a section titled "Triggers" with a sub-section "Actions". The "Triggers" tab is selected. Under "Triggers", there are four items listed:

- On all new alerts (preview) Microsoft Graph Security
- On new high severity alerts (preview) Microsoft Graph Security
- Trigger fires when recommendations are available (preview) ModuleQ
- When a HTTP request is received** Request

The fourth item, "When a HTTP request is received" under "Request", is highlighted with a red rectangular box.

Select **Use sample payload to generate schema**

When a HTTP request is received

HTTP POST URL URL will be generated after save ...

Request Body JSON Schema

Use sample payload to generate schema

Add new parameter ▼

This event publishes when a device is connected to an IoT hub.

Paste the Device connected event schema JSON into the text box, then select Done:

Note: You may receive a pop-up notification that says, Remember to include a Content-Type header set to application/json in your request. You can safely ignore this suggestion, and move on to the next section.

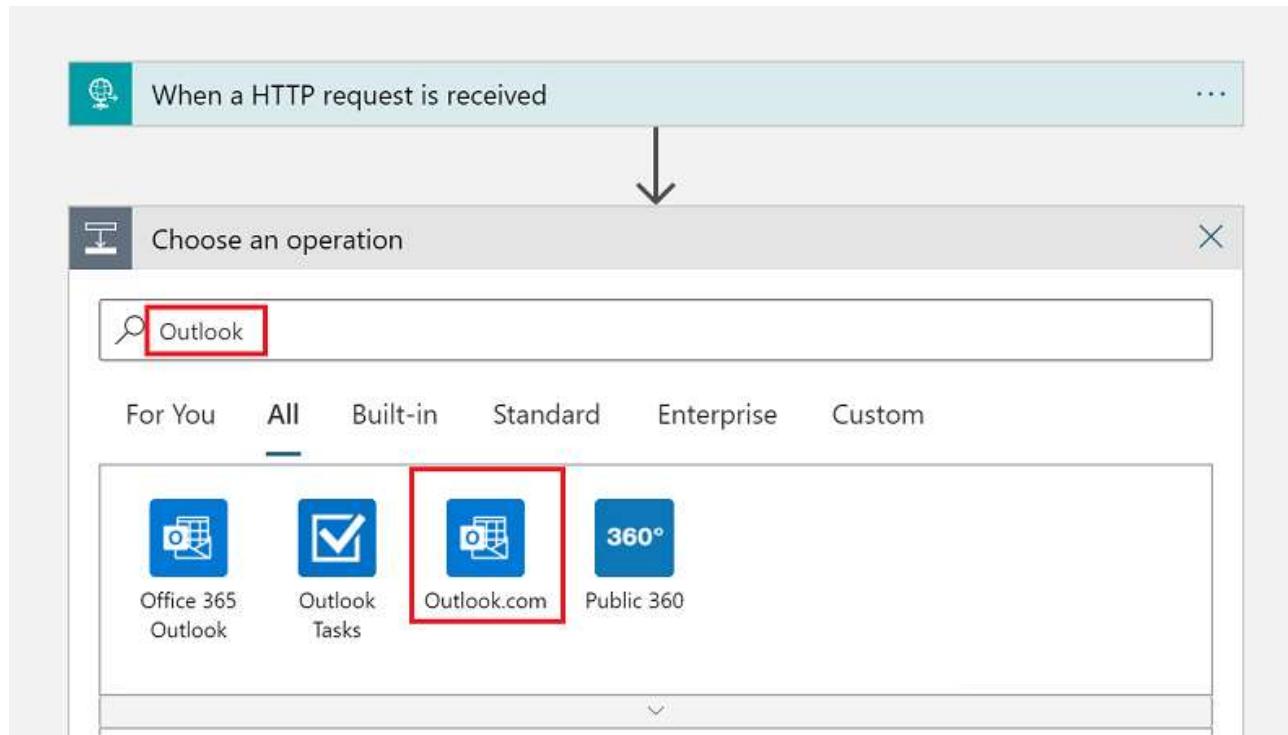
Task 2: Creating the Workflow

Create an Action

Actions are any steps that occur after the trigger starts the logic app workflow. For this tutorial, the action is to send an email notification from your email provider.

Select New step. This opens a window to Choose an action.

Search for Outlook.



Select the **Send an email (V2)** action.

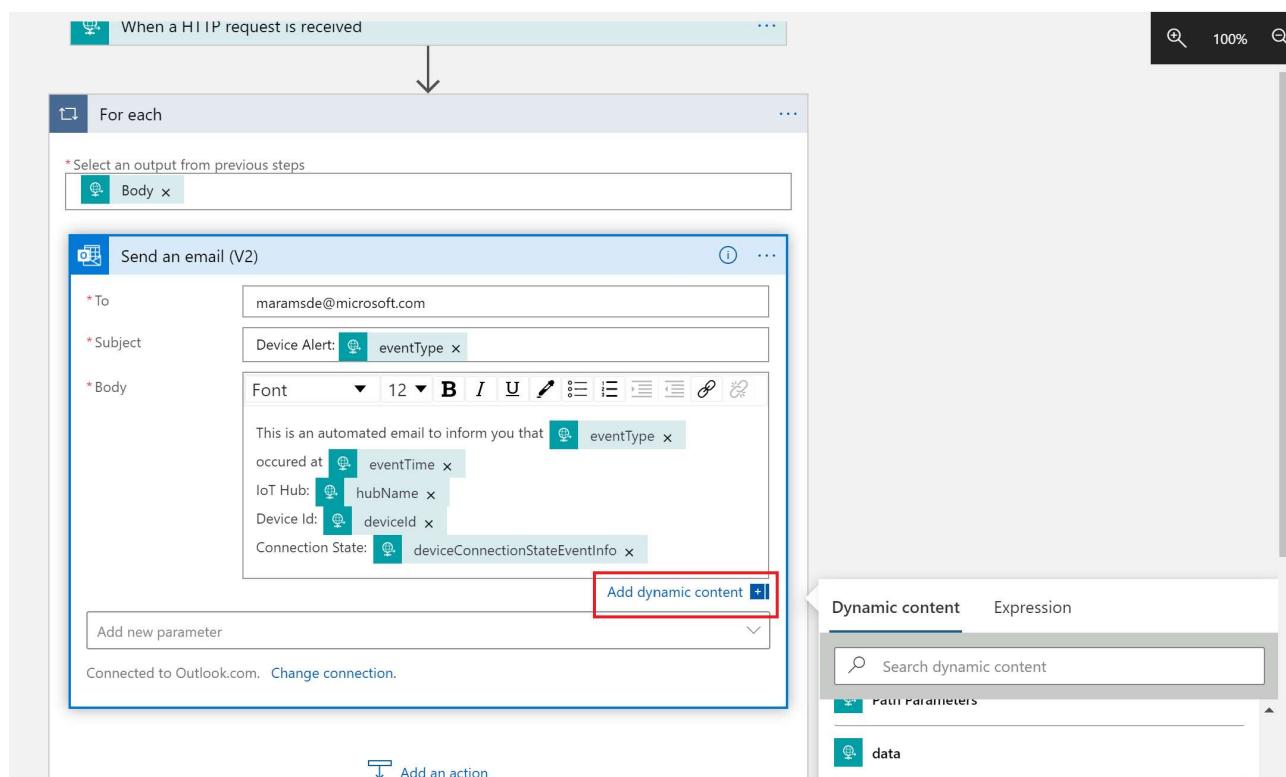
Select Sign in and sign in to your email account. Select Yes to let the app access your info.

Build your email template.

To: Enter the email address to receive the notification emails. For this tutorial, use an email account that you can access for testing.

Subject: Fill in the text for the subject. When you click on the Subject text box, you can select dynamic content to include. For example, this tutorial uses IoT Hub alert: {eventType}. If you can't see Dynamic content, select the Add dynamic content hyperlink -- this toggles it on and off.

Body: Write the text for your email. Select JSON properties from the selector tool to include dynamic content based on event data. If you can't see the Dynamic content, select the Add dynamic content hyperlink under the Body text box. If it doesn't show you the fields you want, click more in the Dynamic content screen to include the fields from the previous action.



Then click **Save** top left of the Logic App.

Copy the HTTP URL

Before you leave the Logic Apps Designer, copy the URL that your logic apps is listening to for a trigger. You use this URL to configure Event Grid.

Expand the When a HTTP request is received trigger configuration box by clicking on it.

Copy the value of HTTP POST URL by selecting the copy button next to it.

The screenshot shows the Azure Logic App Designer interface. On the left, there's a sidebar with various tools: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Development Tools (with Logic app designer selected), Logic app code view, Versions, API connections, Quick start guides, Settings (Workflow settings, Authorization, Access keys), and a 'For each' loop configuration.

The main area displays a trigger configuration titled 'When a HTTP request is received'. It includes an 'HTTP POST URL' field containing the value <https://prod-01.eastus.logic.azure.com:443/workflows/2e4b2b47436e45c7a>. Below it is a 'Request Body JSON Schema' editor showing a JSON structure:

```
{ "items": { "properties": { "data": { "properties": { "deviceConnectionStateEventInfo": { "properties": { "sequenceNumber": { "type": "string" } } } } } } }
```

Below the schema is a link 'Use sample payload to generate schema'. At the bottom, there's a 'Add new parameter' button and a 'For each' loop configuration.

Task 3: Subscribe to the Events

Configure subscription for IoT Hub events

In this section, you configure your IoT Hub to publish events as they occur.

In the Azure portal, navigate to your IoT hub. You can do this by selecting Resource groups, then select the resource group for this tutorial, and then select your IoT hub from the list of resources.

Select Events.

The screenshot shows the Azure IoT Hub Overview page for the hub 'My-hub-mpr'. The sidebar on the left includes: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems (with Events highlighted with a red box), and Settings.

Select + Event subscription

Complete the following form:



Create Event Subscription

Event Grid

Basic Filters Additional Features

Event Subscriptions listen for events emitted by the topic resource and send them to the endpoint resource. [Learn more](#)

EVENT SUBSCRIPTION DETAILS

Name *

Event Schema

Event Grid Schema

TOPIC DETAILS

Pick a topic resource for which events should be pushed to your destination. [Learn more](#)

Topic Type



IoT Hub

Source Resource

My-hub-mpr

System Topic Name



EventsTopic

EVENT TYPES

Pick which event types get pushed to your destination. [Learn more](#)

Filter to Event Types

Device Disconnected



Device Created

Device Deleted

Device Connected

Device Disconnected

Device Telemetry

ENDPOINT DETAILS

Pick an event handler to receive your events

Endpoint Type *

- **Name:** devicestate
- **Event Schema:** Event Grid Schema
- **Topic Type:** Your IoT Hub created during this training should appear as default
- **System topic Name:** EventsTopic
- **Event Types :** Select only Device Disconnected
- **Endpoint Type:** Select Webhook
- **Endpoint:** Select an Endpoint, in the new window paste the URL from the Logic App created in previous step. Then Confirm Selection.

Select Create

Exercise 6: Interacting with remote devices

Task 1: Invoke a direct method

In the Azure portal, invoke the method with the method name RestartModule and the following JSON payload.

Open two tabs with Azure Portal, start monitoring your device payload so you can see when the device is receiving the reboot request, like you did in Exercise #4, Task #1.

Nex, in your IoT Hub open the Edge device, in **Modules** tab at the bottom look for the **\$edgeAgent** Module, click on it </>Direct Method

The screenshot shows the 'IoT Edge Module Details' page for the '\$edgeAgent' module. At the top, there are tabs for 'Module Identity Twin' (selected), 'Direct method' (highlighted with a red box), and 'Refresh'. Below these, there are fields for 'Module Identity Name' (set to 'my-device-mpr/\$edgeAgent'), 'Primary key', 'Secondary key', 'Connection string (primary key)', and 'Connection string (secondary key)', each containing a long string of asterisks. At the bottom of the page, there are links for 'IoT Edge Module Settings', 'Container Create Options', and 'Environment Variables'.

1. Method Name: **SimulatedTemperatureSensor**
2. Copy the following request in the **Payload** section

```
{  
  "schemaVersion": "1.0",  
  "id": "SimulatedTemperatureSensor"  
}
```

3. Click on **Invoke Method**



Direct method



...

my-device-mpr/\$edgeAgent

| Directory: Microsoft

Invoke Method

3



You can use this tool to send direct methods to a module in your hub.

Module Identity

my-device-mpr/\$edgeAgent

Method Name

1

RestartModule

Payload

```
{  
    "schemaVersion": "1.0",  
    "id": "SimulatedTemperatureSensor"  
}
```

2

Connection Timeout



Method Timeout



Result ⓘ

```
{"status":200,"payload":null}
```

4

4. Check the results, you should receive a 200 Status as a reboot successfully occurs.

Task 2: Invoke a direct method, checking the logs

Once you run the module reboot, your telemetry data was stopped for a while, you should be able to see this pause while seeing the payload running in CLI.

Once you invoke a direct method to retrieve the logs, you should be able to see the reboot requested before

Fill the form again requesting the logs as follow

1. Method Name: **GetModuleLogs**
2. Copy and Paste the following json the **Payload** section

```
{
    "schemaVersion": "1.0",
    "items": [
        {
            "id": "edgeAgent",
            "filter": {
                "tail": 10
            }
        }
    ],
    "encoding": "none",
    "contentType": "text"
}
```

3. Click **Invoke Method** to see the results
4. Check the results you should see the reboot you sent in previous task.

Direct method

my-device-mpr/\$edgeAgent | Directory: Microsoft

Invoke Method 3

You can use this tool to send direct methods to a module identity. Direct methods have a name, payload, and configurable connection and timeout settings.

Module Identity ⓘ
my-device-mpr/\$edgeAgent

Method Name ⓘ 1
GetModuleLogs

Payload ⓘ 2

```
{
  "id": "edgeAgent",
  "filter": {
    "tail": 10
  },
  "encoding": "none",
  "contentType": "text"
}
```

Connection Timeout ⓘ

Method Timeout ⓘ

Result ⓘ 4

```
{"status":200,"payload":[{"id":"edgeAgent","payloadBytes":null,"payload":"<6> 2021-02-12 18:50:11.925 +00:00 [INF] - Starting periodic op\n18:50:11.926 +00:00 [INF] - Scraping endpoint http://edgeHub:9600/metrics\n<6> 2021-02-12 18:50:11.927 +00:00 [INF] - Scraping endpoint\n02-12 18:50:11.951 +00:00 [INF] - Scrapped and Stored Metrics\n<6> 2021-02-12 18:50:11.951 +00:00 [INF] - Successfully completed periodic task\nGetModuleLogs\n<6> 2021-02-12 19:43:04.105 +00:00 [INF] - Received request GetModuleLogs with payload\n<6> 2021-02-12 19:43:04.105 +00:00 [INF] - Request payload: {\n  \"id\": \"edgeAgent\",\n  \"filter\": {\n    \"tail\": 10,\n    \"since\": null,\n    \"until\": null,\n    \"logLevel\": null,\n    \"regex\": null\n  },\n  \"encoding\": \"0\",\n  \"contentType\": \"1\"\n}"}]
```

Exercise 7: Clean up

Task 1: Delete resources

After the hands-on lab

In this exercise, you will delete any Azure resources that were created in support of the lab. You should follow all steps provided after attending the Hands-on lab to ensure your account does not continue to be charged for lab resources.

Delete the resource group or all the resources within the resource group.

Using the Azure portal, navigate to the Resource group you used throughout this hands-on lab by selecting Resource groups in the left menu.

Search for the name of your research group and select it from the list.

Select Delete in the command bar and confirm the deletion by re-typing the Resource group name, and selecting Delete.

You should follow all steps provided after attending the Hands-on lab.

Target audience

IoT Academy Atendees

Hands-on lab

At the end of this hands-on lab, you will be better able to understand core services when building an IoT Solution.

Azure services and related products

- VS Code, installed locally [Download here](#).
- Putty, installed locally, [Download here](#).
- Azure IoT Hub
- CLI
- Ubuntu Server, Virtual Machine
- Logic App
- Event Grid
- Stream Analytics
- Storage Account

Azure solution

Internet of Things