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

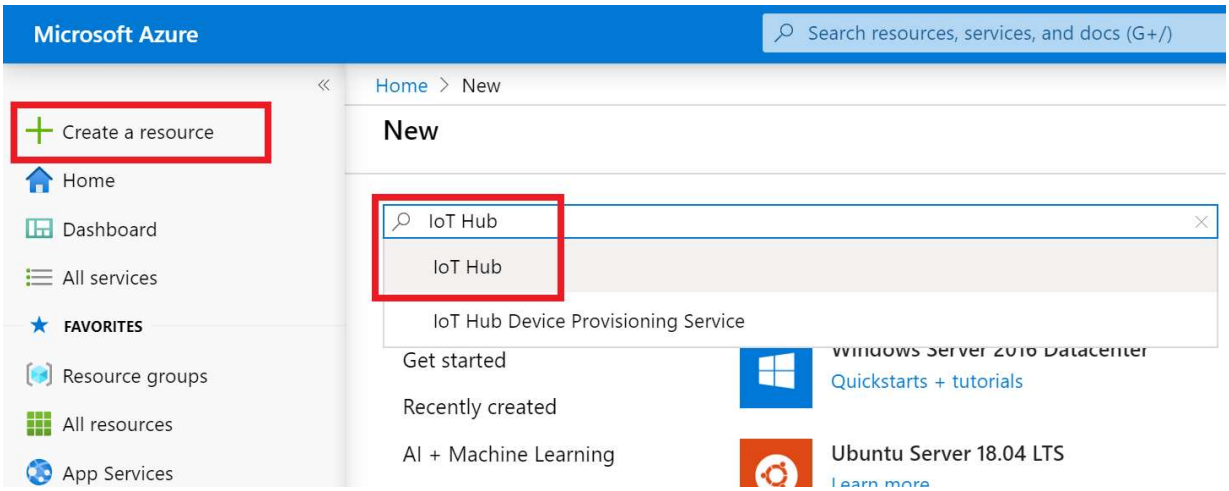
Exercise 1: IoT Hub provisioning

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.

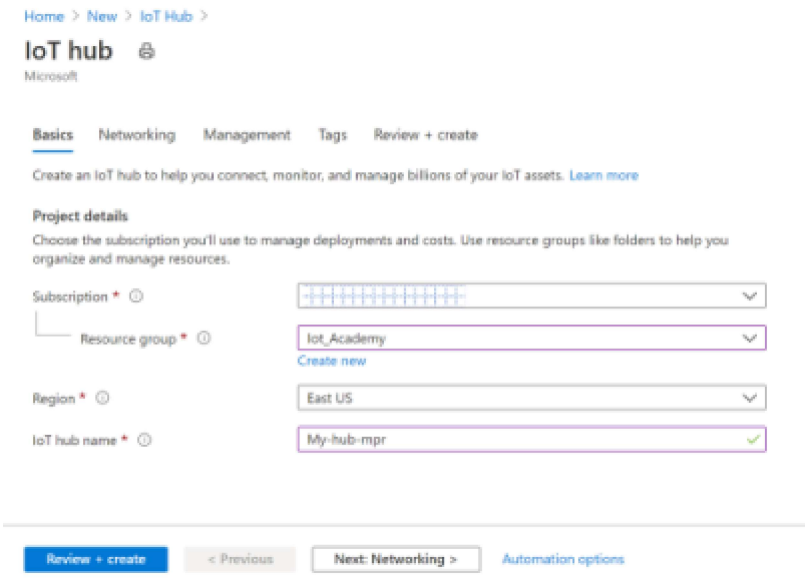
1. In your browser, navigate to the Azure portal, select **+Create a resource** in the navigation pane, enter `iot` into the **Search the Marketplace** box.

2. Select **IoT Hub** from the results, and then select **Create**.



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

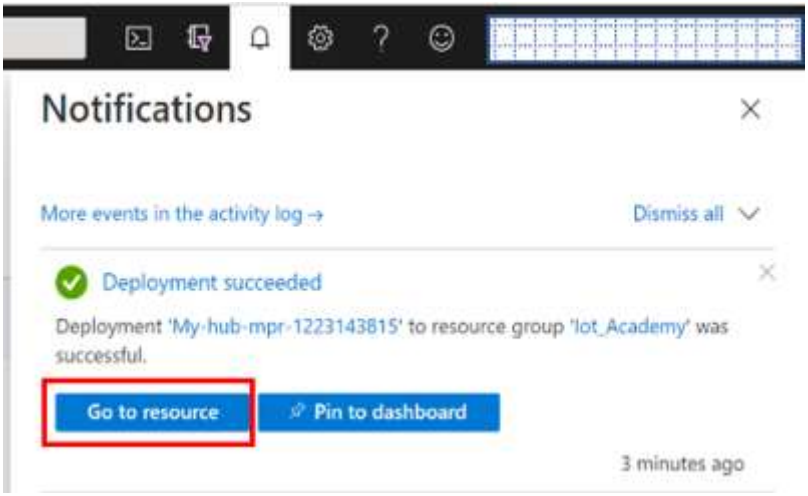
   - **Subscription**: Select the subscription you are using for this hands-on lab.
   - **Resource group**: Choose Use existing and select the resource group.
   - **Region**: Select the location you are using for this hands-on lab.
   - **IoT Hub Name**: Enter a unique name, such as `my-hub-SUFFIX`.



   - 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**.

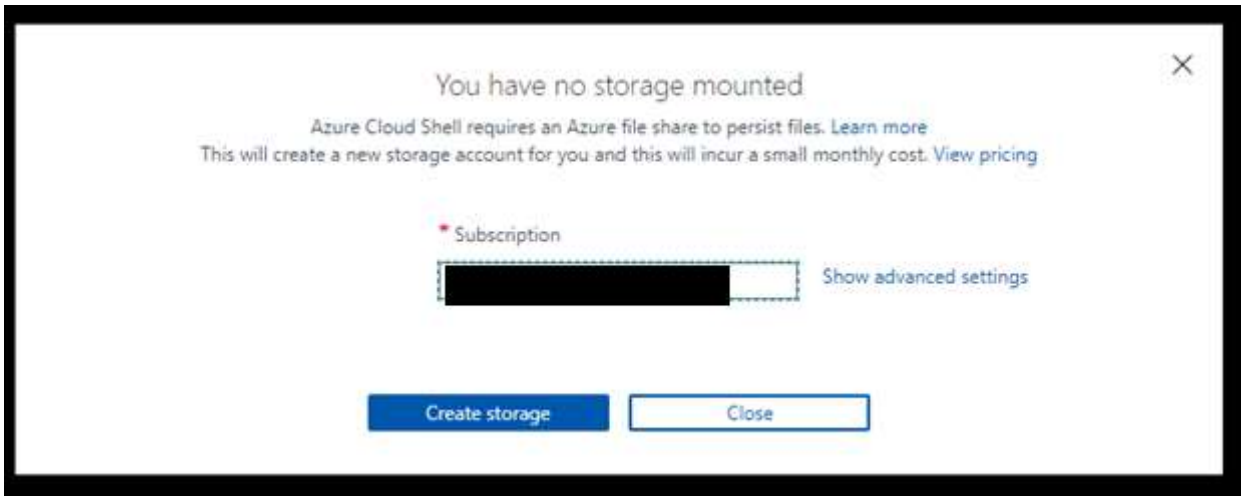
- Select **Create** on the **Review + create** blade.

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

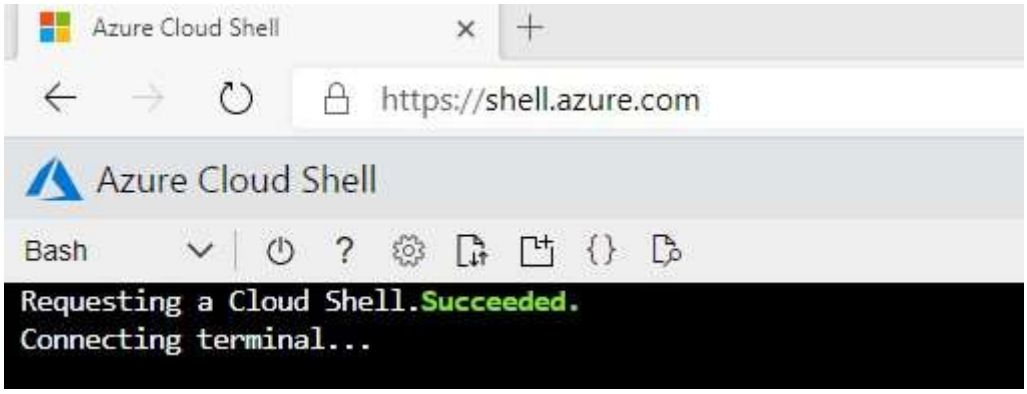
If you never used before you will be prompted to mount an storage account, click **Create Storage** to continue. If you used before, you will skip this step.



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:

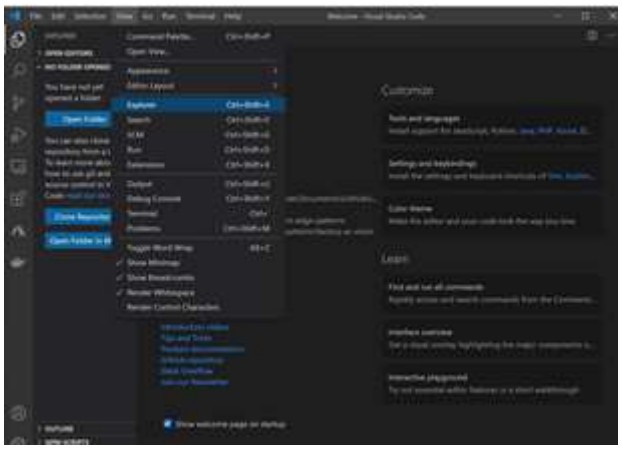
```
az iot hub delete --name {your iot hub name}  --resource-group {your resource group
```

Task 3: Provision IoT Hub through VS Code

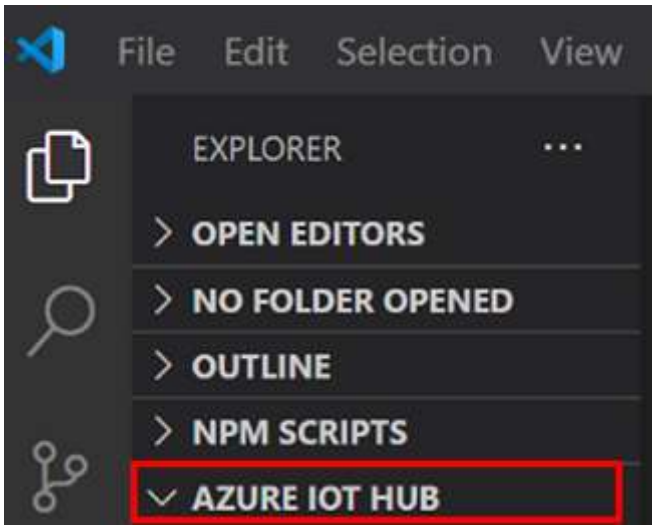
Last we will use another tool to also create an IoT Hub, in this case VS Code. For this task make sure you download VS Code in advance. Download link: https://code.visualstudio.com/Download

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

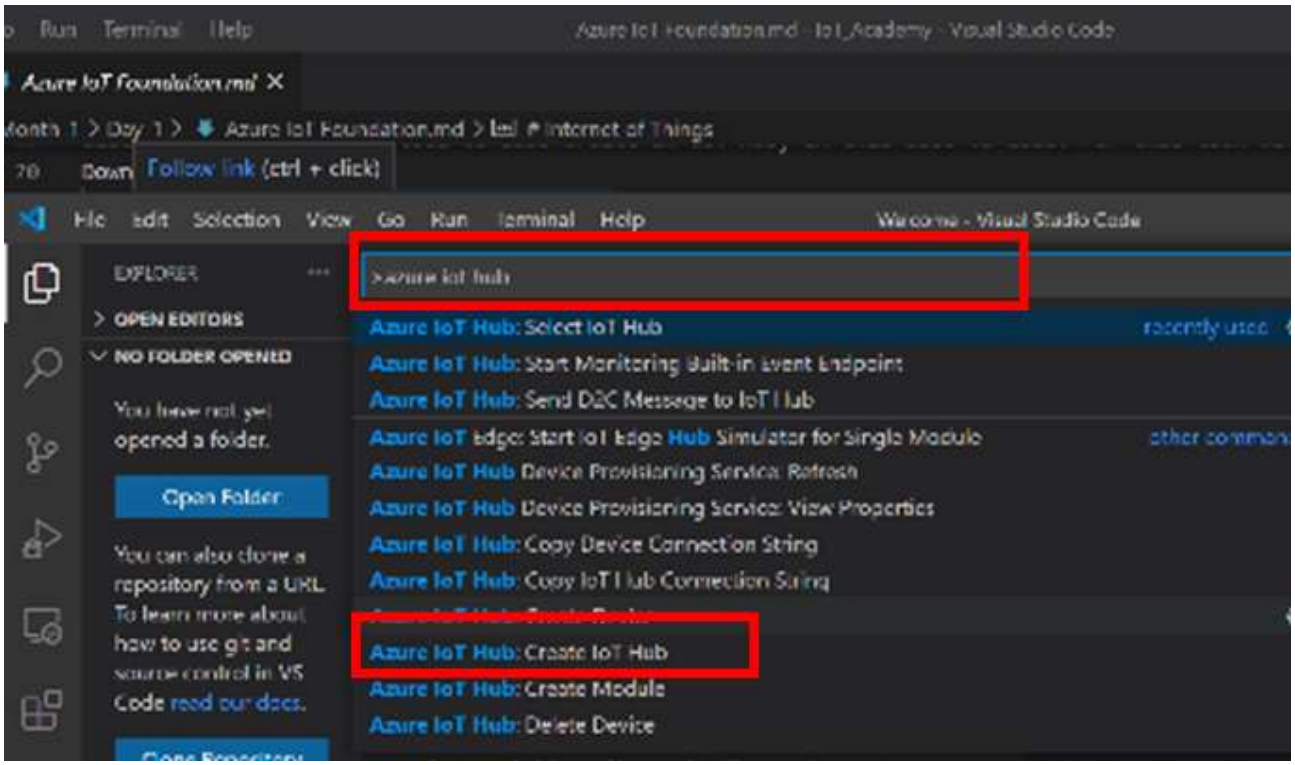
2. Go To View Explorer



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 IoTHub `vscodeiotcreate` add suffix as needed.

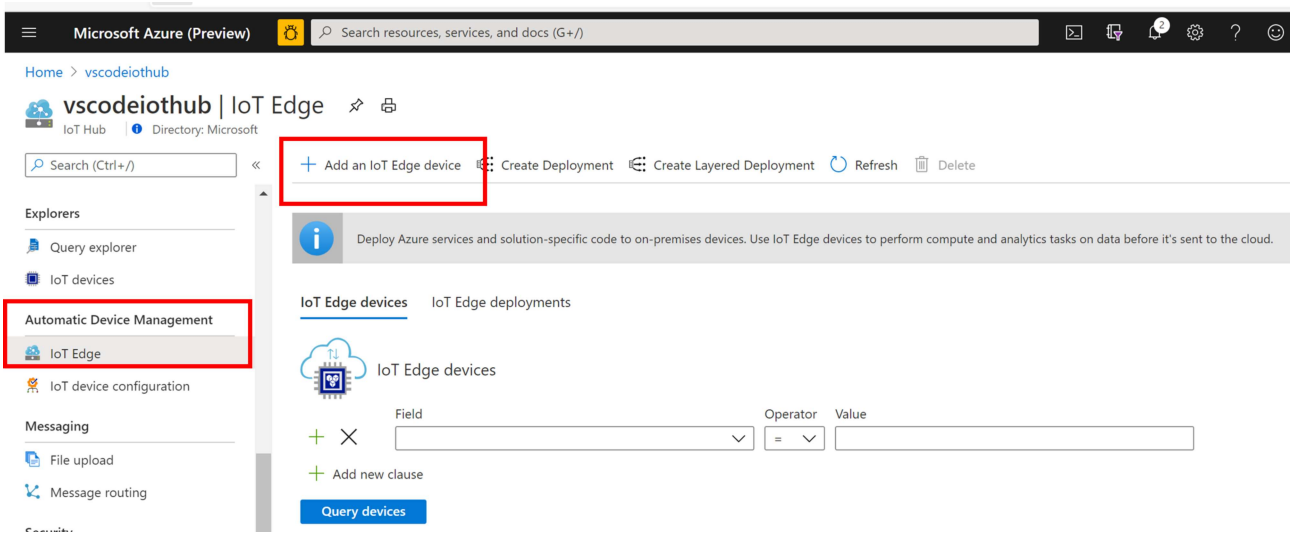
After the creation process you should be able to see the new IoT Hub in Azure Portal and in VS Code.

Exercise 2: Devices

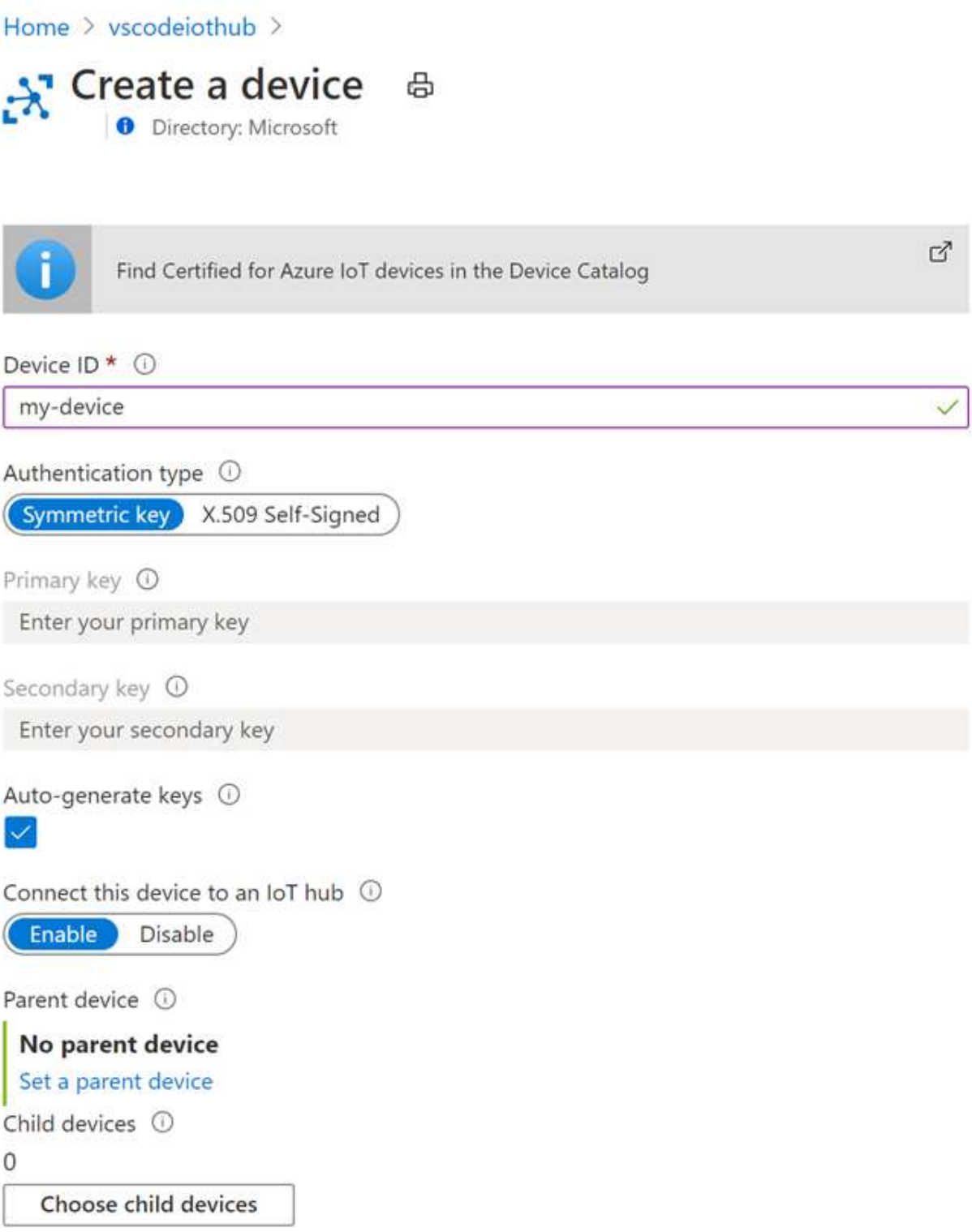
During this exercise you will learn how to set up and edge device and connect your device to IoT Hub to start streaming data.

Task 1: Setting up a Device

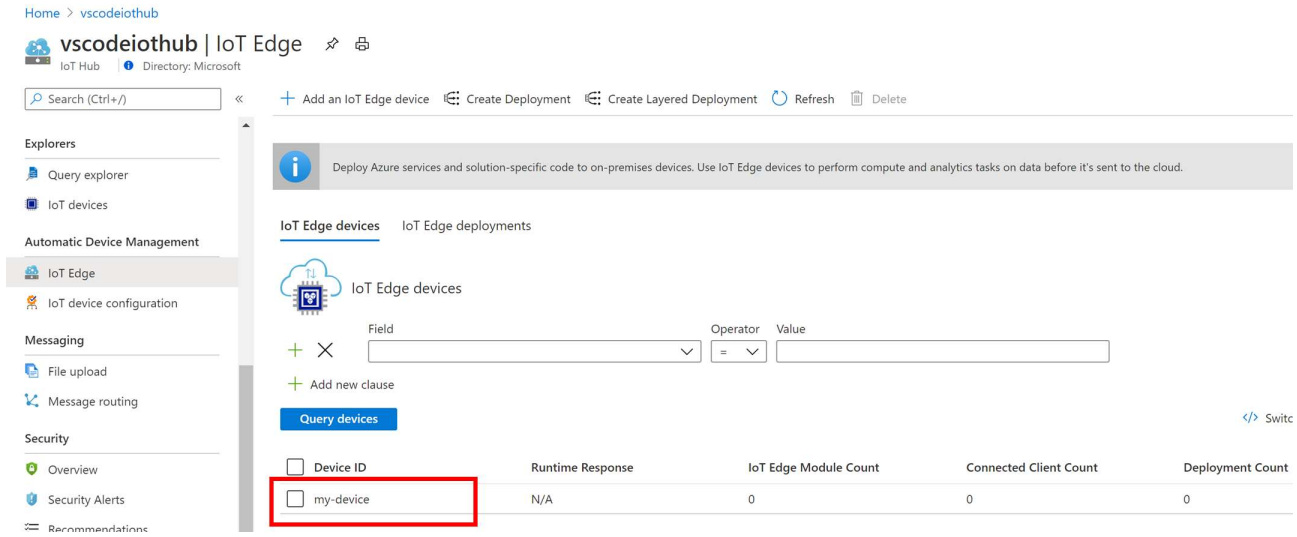
From Azure Portal select the IoT Hub created through VS Code previously, scroll down to **Automatic Device Management** then select **Add an IoT Edge device**



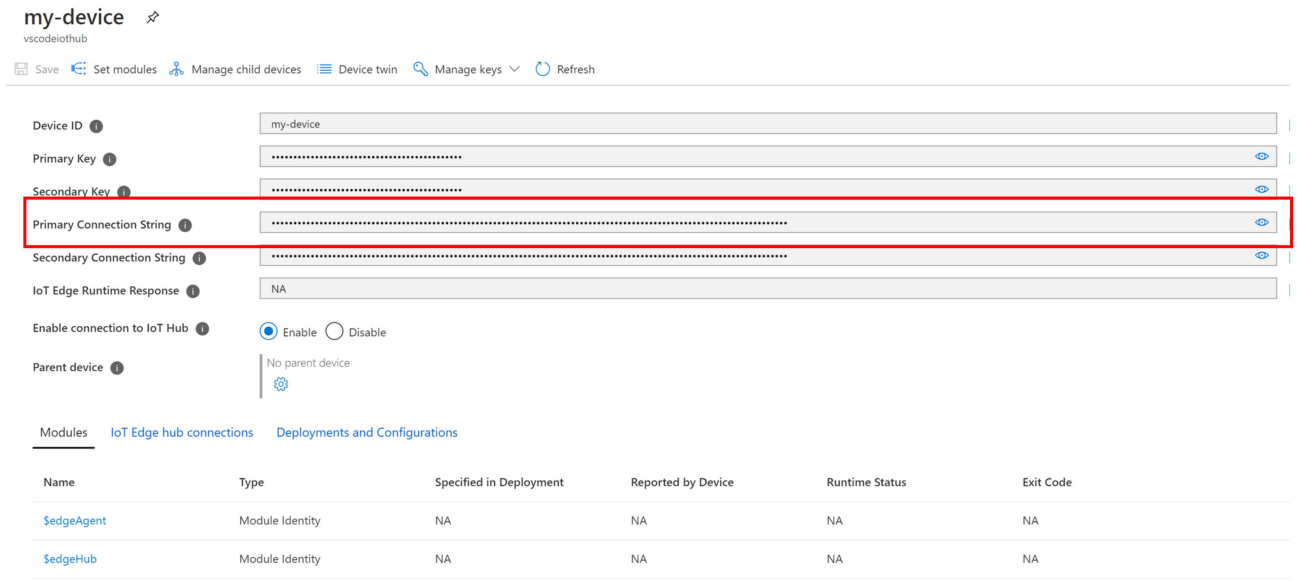
In the new window select a name for your device **my-device-SUFFIX** and click **Save**



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



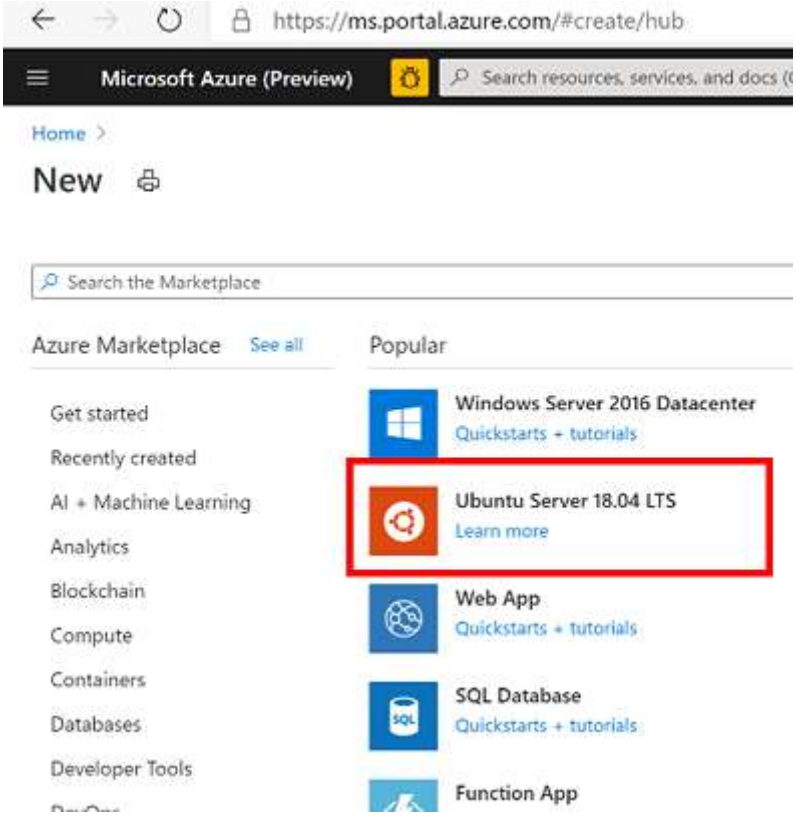
Now, copy and paste in the a notepad the connection string of your device, you will need this to connect your device to IoT hub, we will use this connection string in the next task.



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.



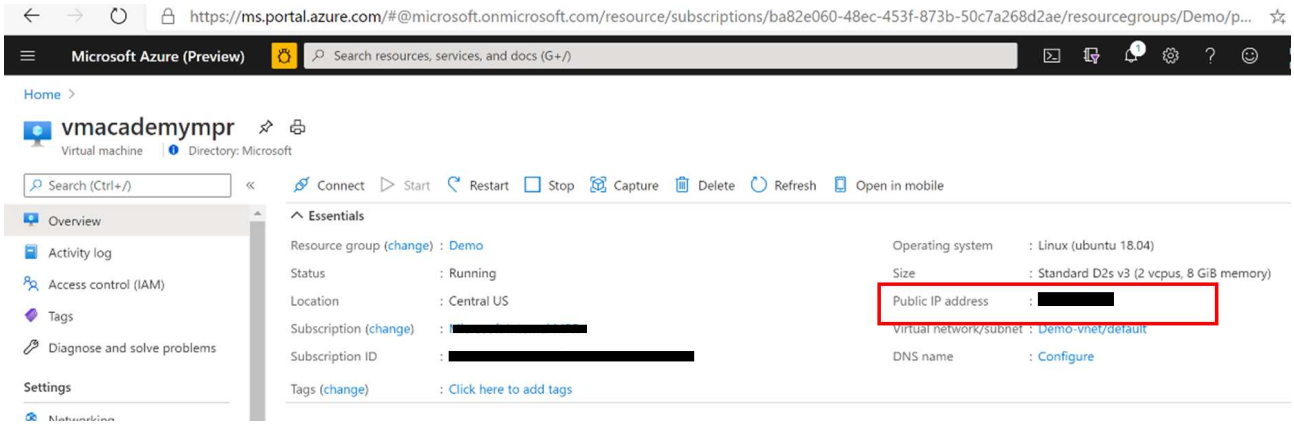
2. Then you will need to complete the following parameters in the **Basics** tab:

    - **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

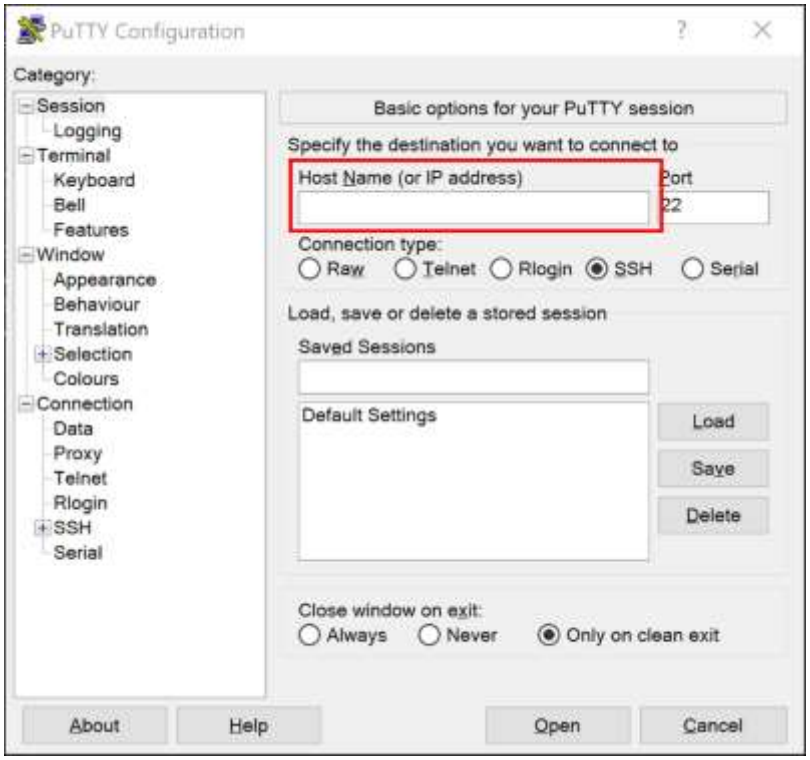
    In **Management** Tab disable monitoring for this lab.

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

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



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 doesnt 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 successfull 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 > ./micr
```

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.g
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
```

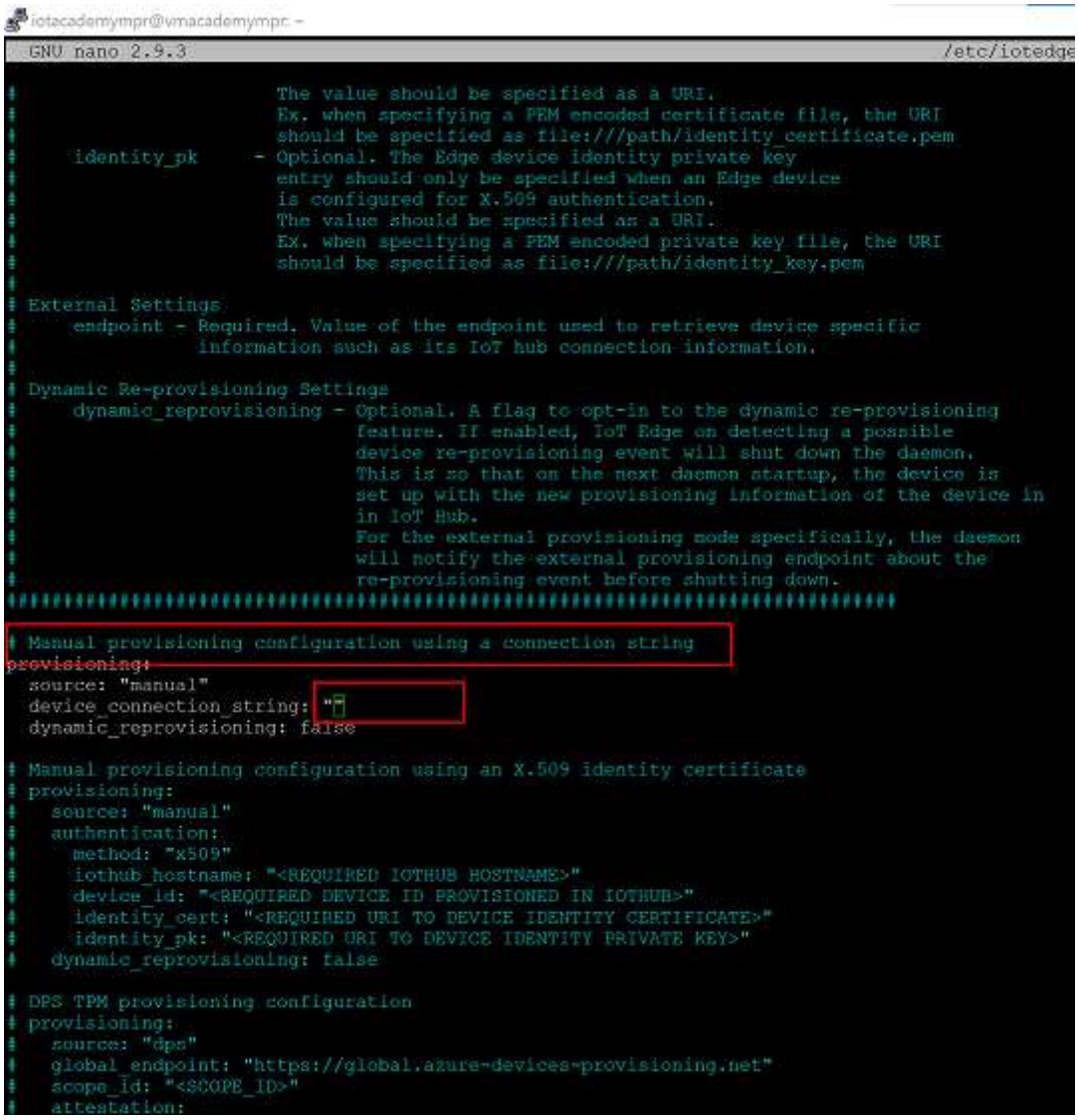
If you want to install the most recent version of the security daemon, use the following command that also installs the latest version of the libiothsm-std package:

```
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 wit the connection string from the task 1.



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
```

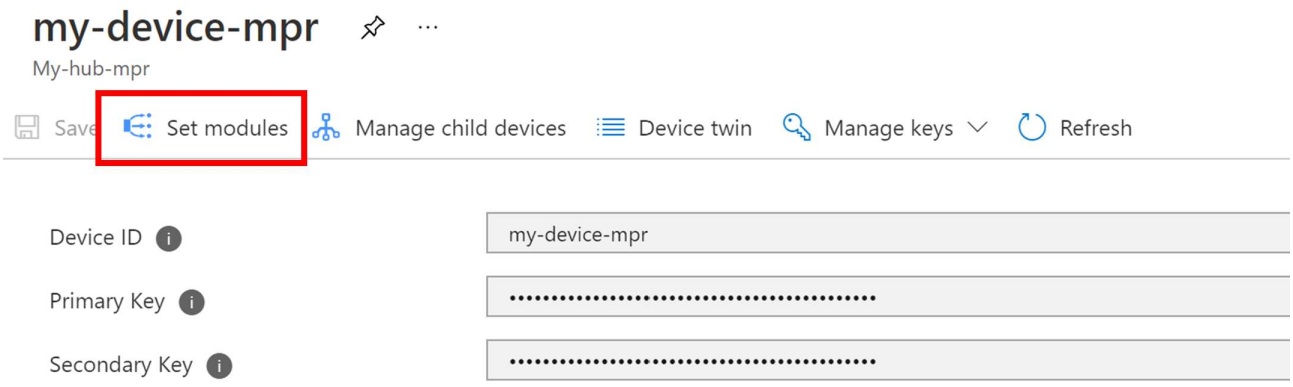
In a few minutes you should receive a **Running** Status after executing the following command:

```
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**



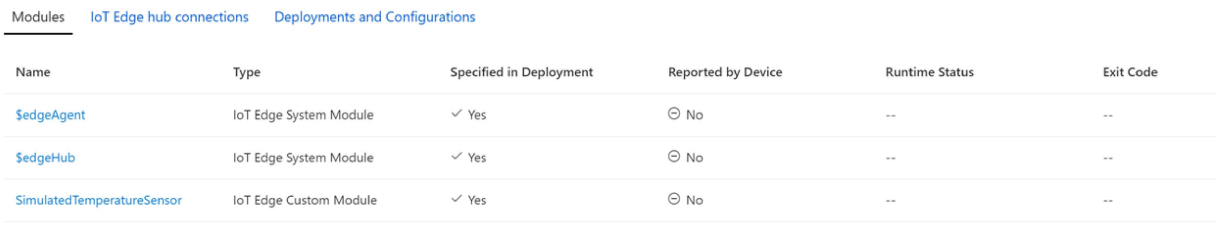
Then to add the temperature simulator, select **Add** then you will see multiple options, select **+ Marketplace Module**

The search bos 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.

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

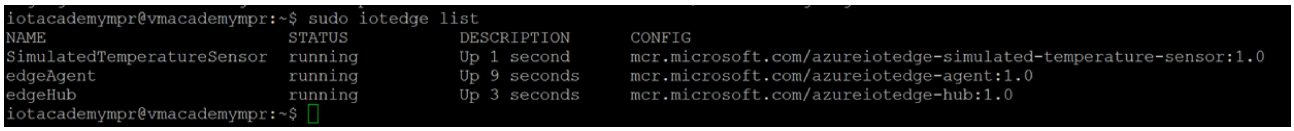
- Check the modules running in your Edge Device:



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

```
sudo iotedge list
```

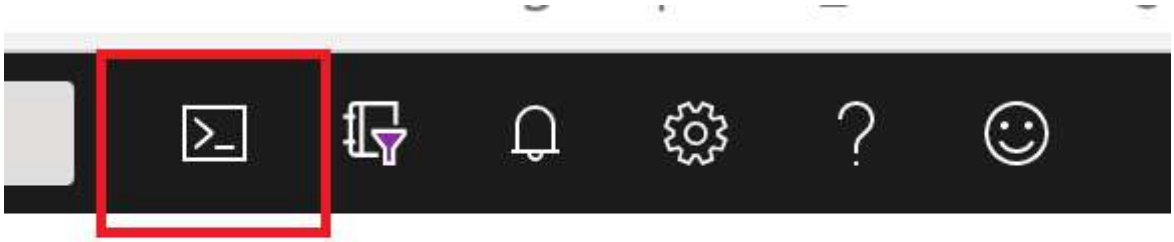
Now you should see all the modules up and running in your Ubuntu Machine as shown below:



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:



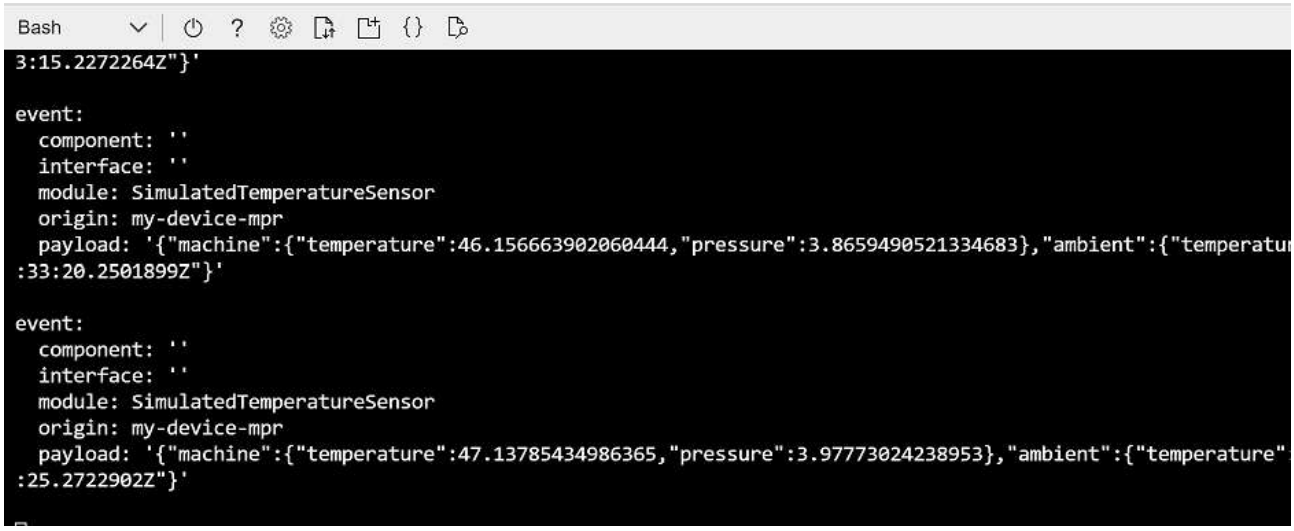
Run the az extension add command to add the Microsoft Azure IoT Extension for Azure CLI to your CLI shell. The IOT Extension adds IoT Hub, IoT Edge, and IoT Device Provisioning Service (DPS) specific commands to Azure CLI.

```
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:

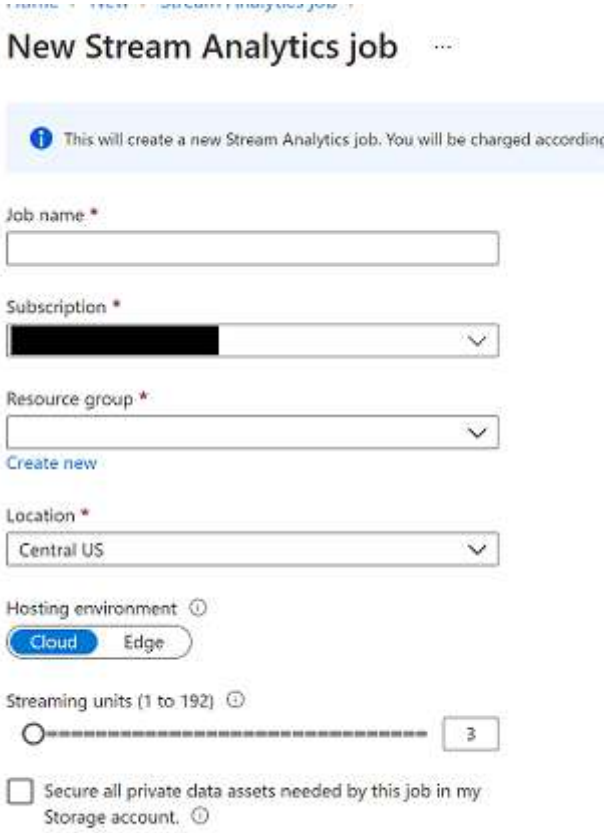


Task 2: Setting up Services

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

Fill the fields in the form:



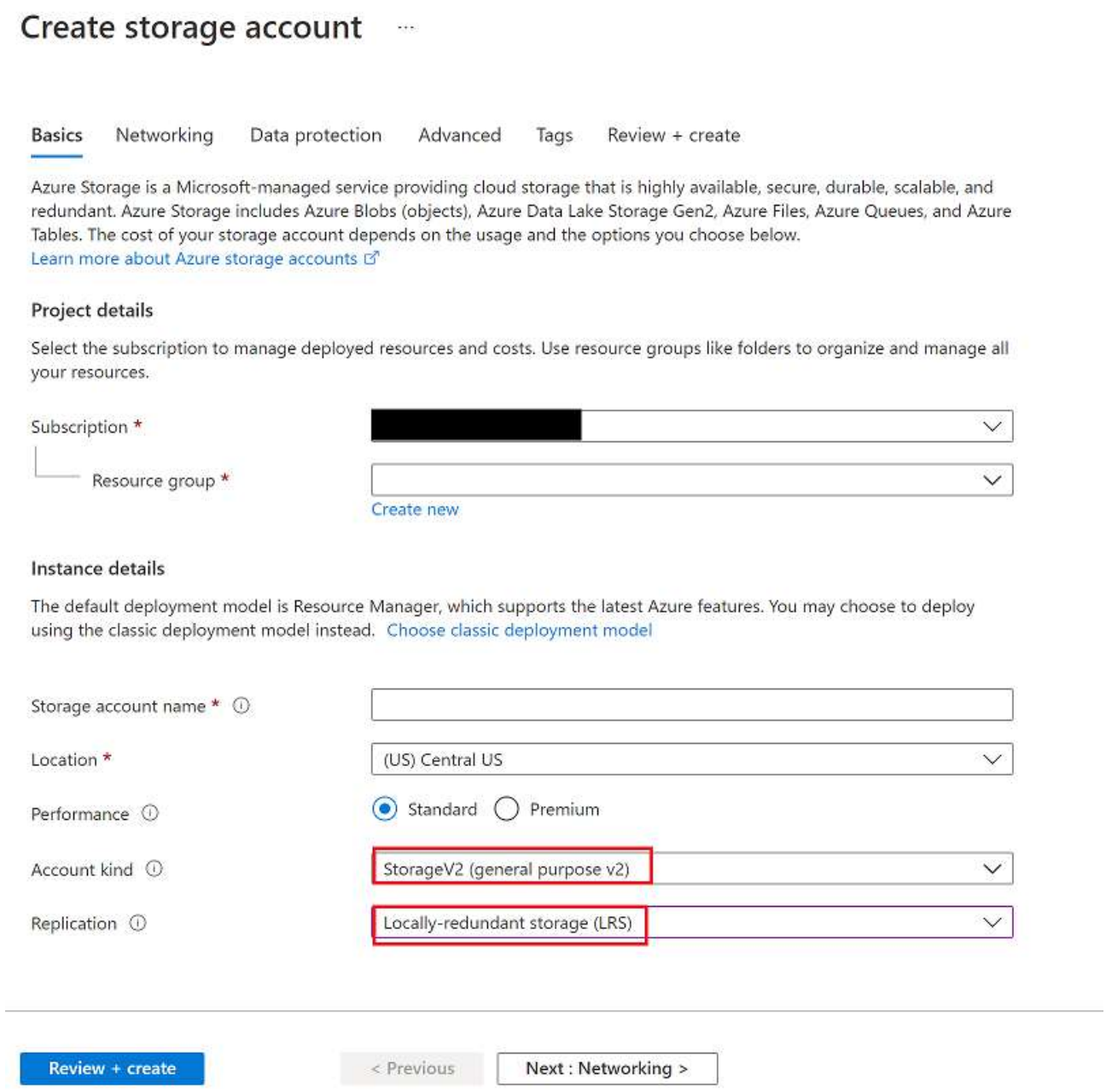
- **Job name**: satrainingSUFFIX
- **Subscription**: Select The subscription you are using for this training
- **Resource Group**: Select the Resource Group you are using for this training.
- **Location**: Select the Location you are using for this training.
- **Hosting Environment**: Cloud
- **Streaming 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** buttom in Azure Portal, in the search box type **Storage Account**, then **Create** again.

Fill the fields in the form:



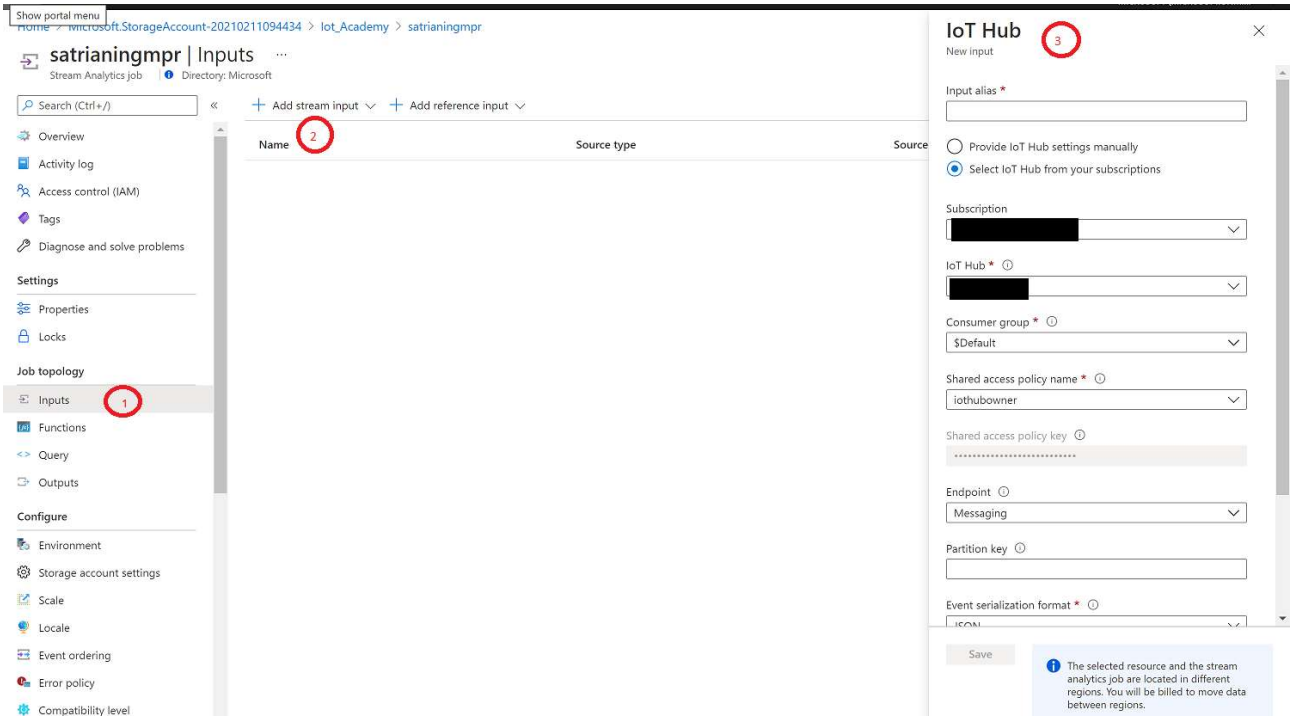
Fill the fields in the form:

- **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**.

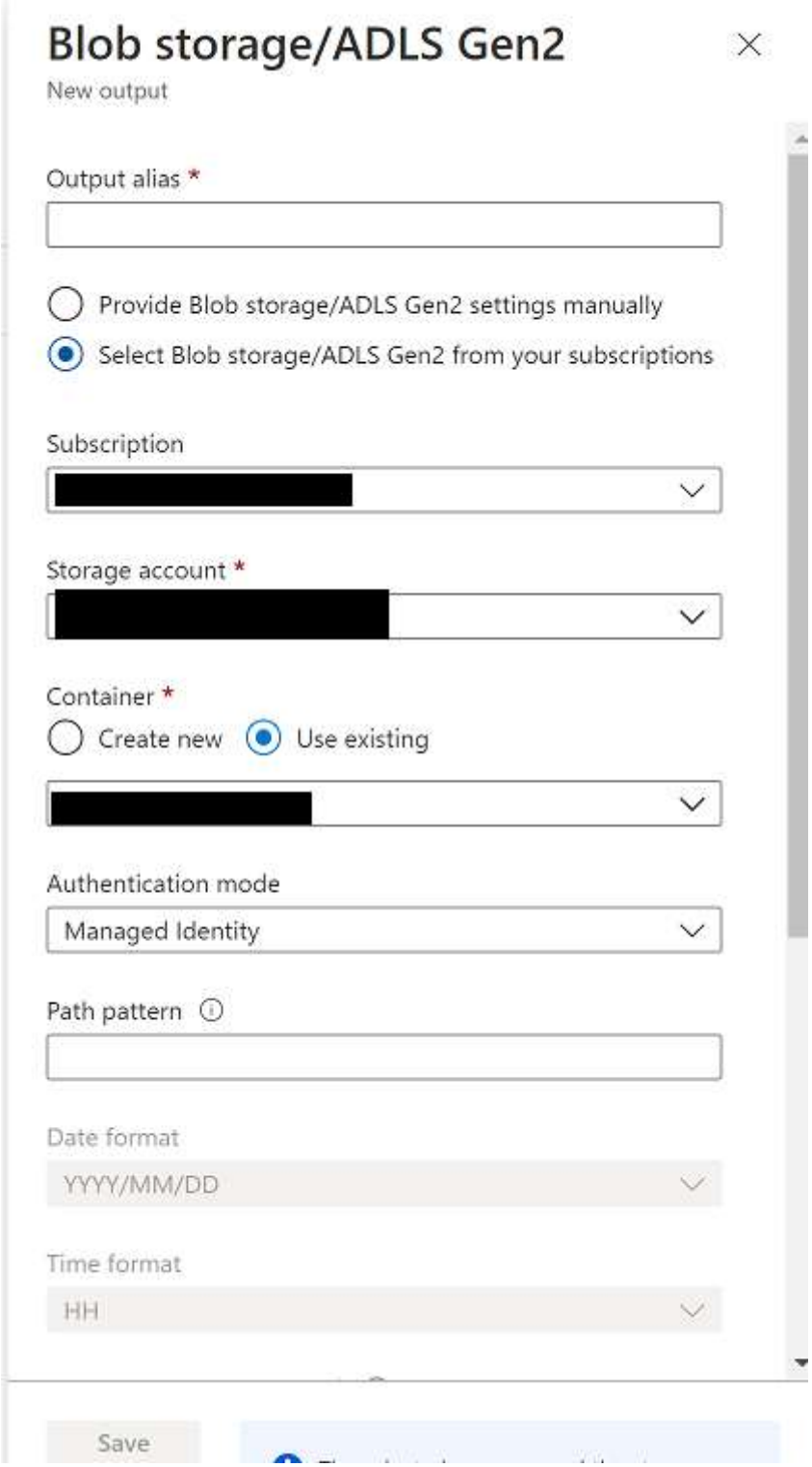
Task 3: Connecting the services

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**.



- **Input Aalias**: inputhub
- 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 defaul 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**



- **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 wit the Input and Output defined in previous steps:

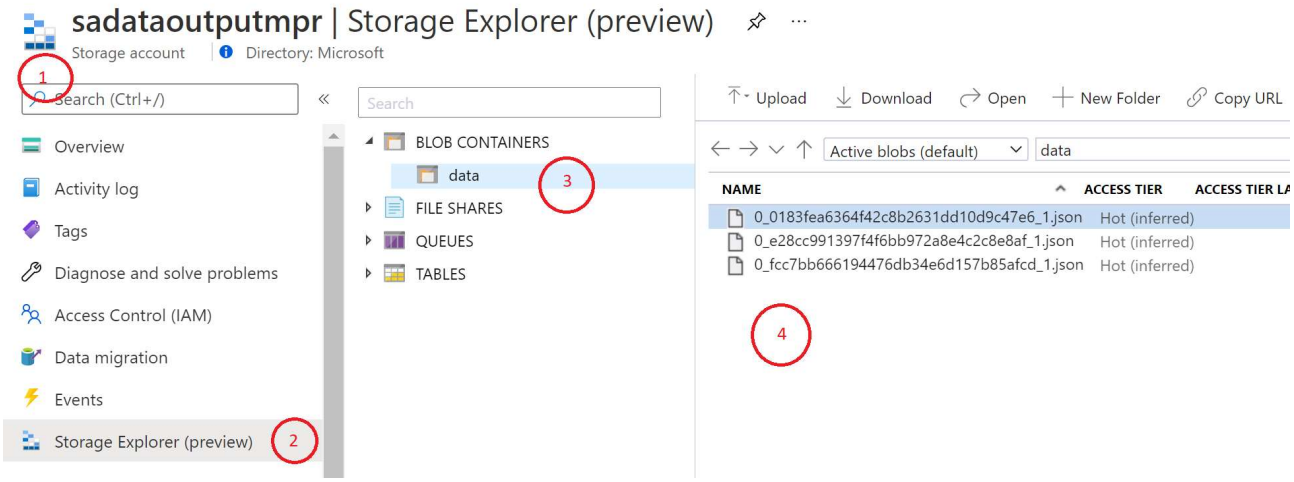
Copy the following command in the window"

```
SELECT
    *
INTO
    storageaccountdata
FROM
    inputhub
```

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, dowloand to see it.

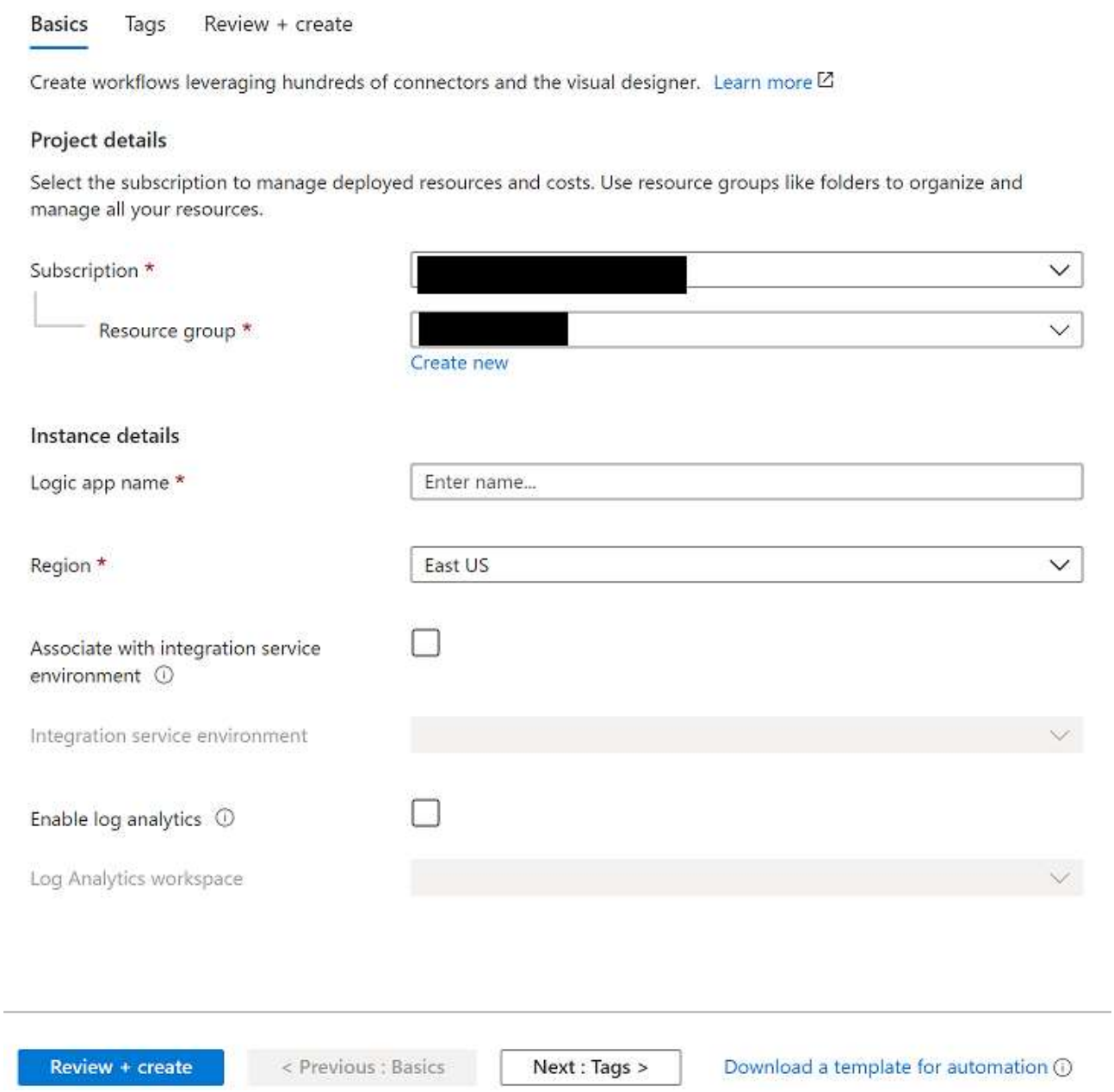


Exercise 5: Monitoring Remote Devices

Task 1: Setting up alerts

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

Fill in the fields in the form:



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.

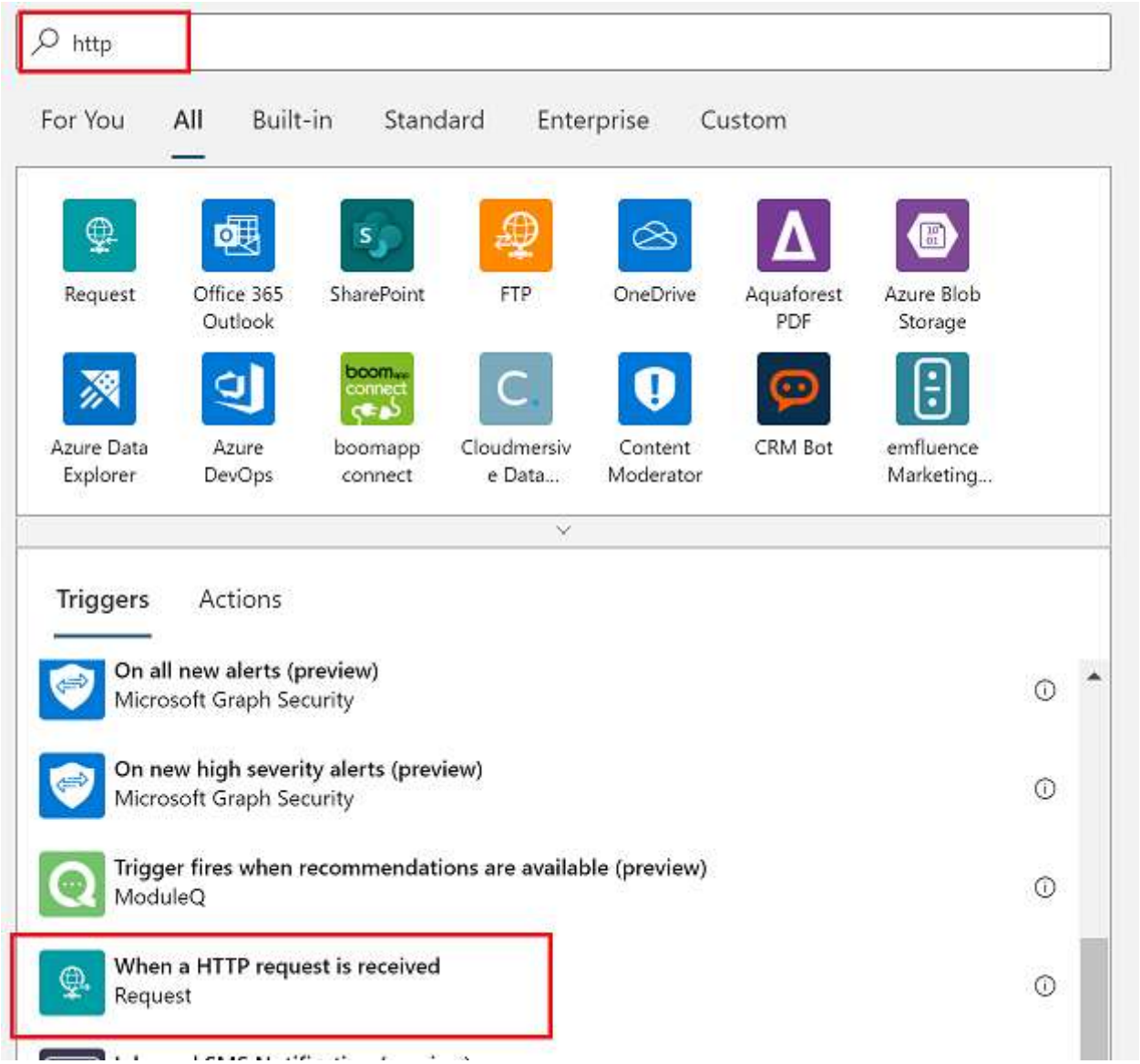
Keep the remaining selections as default. Then select **Review + Create**, after succesfull validation click **Create**

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

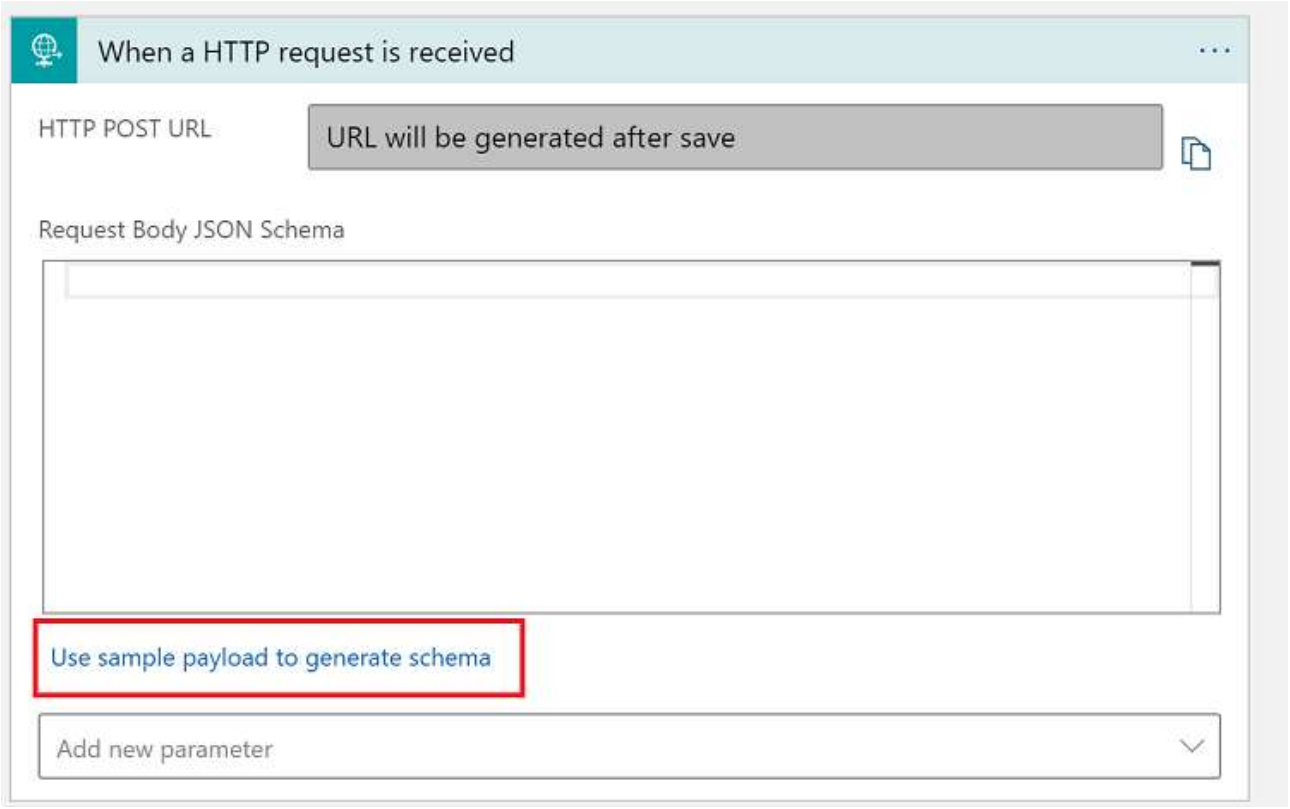
A trigger is a specific event that starts your logic app. For this tutorial, the trigger that sets off the workflow is receiving a request over HTTP.

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

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



Select **Use sample payload to generate schema**



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:

```
[{
   "id": "f6bbf8f4-d365-520d-a878-17bf7238abd8",
   "topic": "/SUBSCRIPTIONS/<subscription ID>/RESOURCEGROUPS/<resource group name>/P
   "subject": "devices/LogicAppTestDevice",
   "eventType": "Microsoft.Devices.DeviceConnected",
   "eventTime": "2018-06-02T19:17:44.4383997Z",
   "data": {
       "deviceConnectionStateEventInfo": {
         "sequenceNumber":
           "000000000000000001D4132452F67CE20000000200000000000000000000000001"
       },
     "hubName": "egtesthub1",
     "deviceId": "LogicAppTestDevice",
     "moduleId" : "DeviceModuleID"
   },
   "dataVersion": "1",
   "metadataVersion": "1"
 }]
```

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.

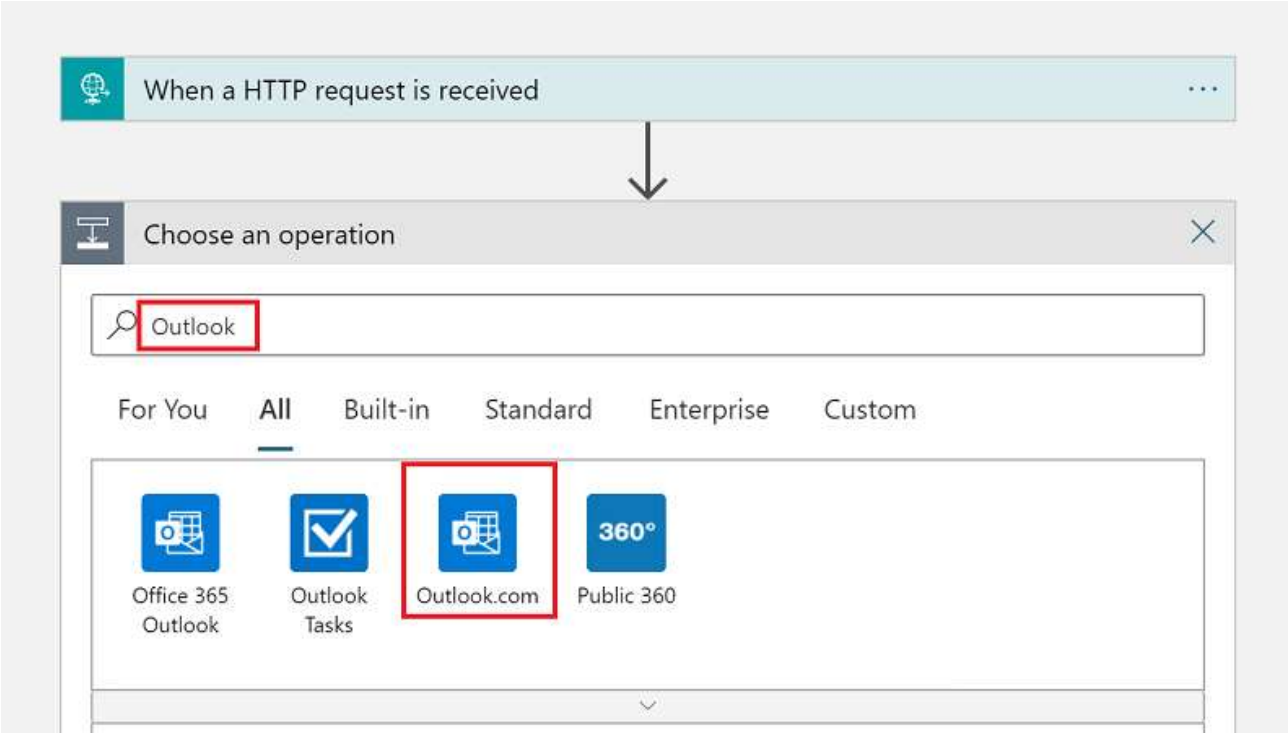
Task 2: Creating the Workflow

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.

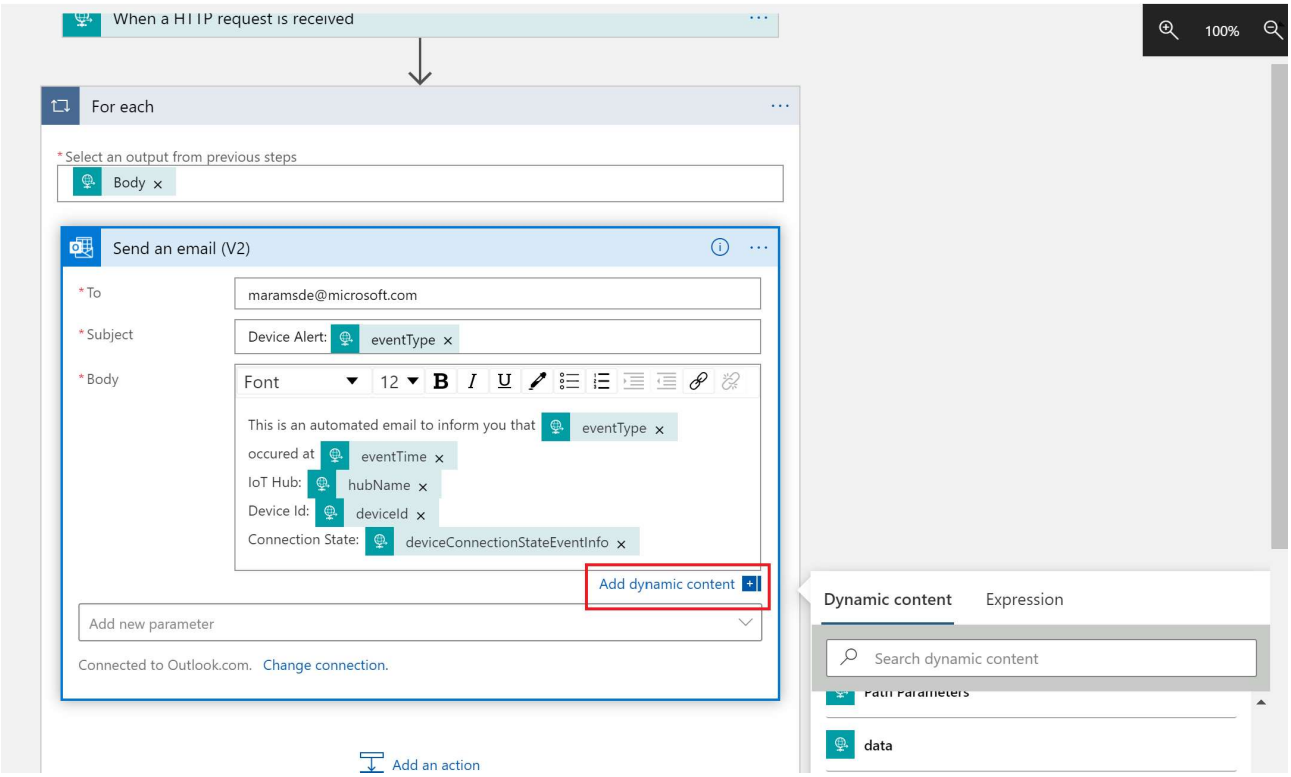
Select Sign in and sign in to your email account. Select Yes to let the app access your info.

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.



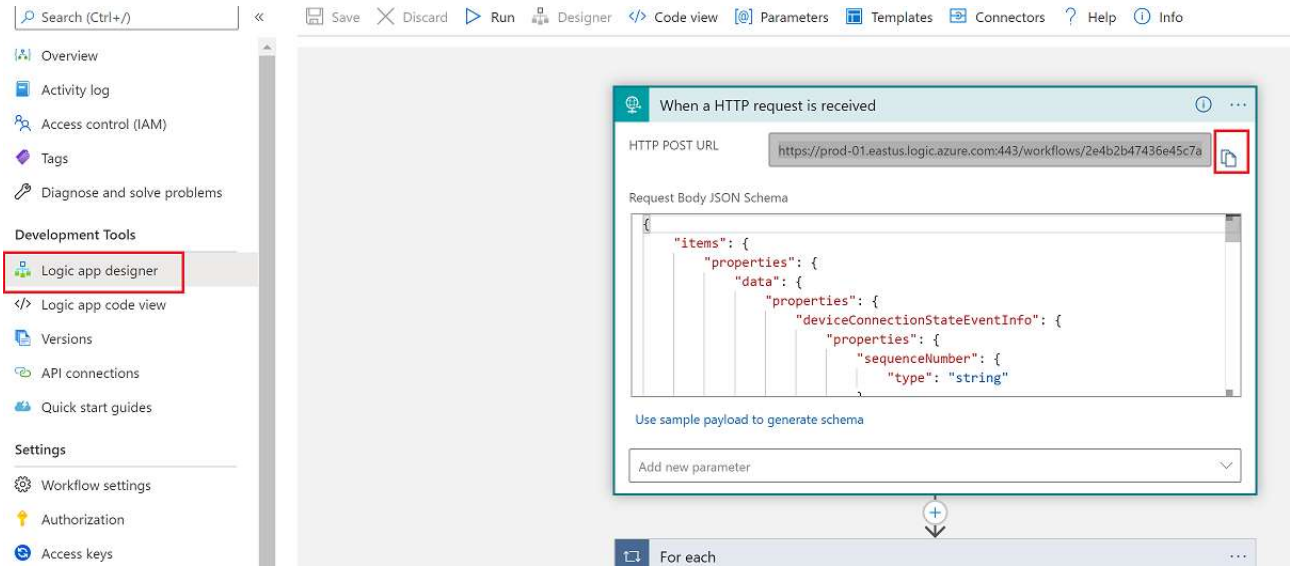
Then click **Save** top left of the Logic App.

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.



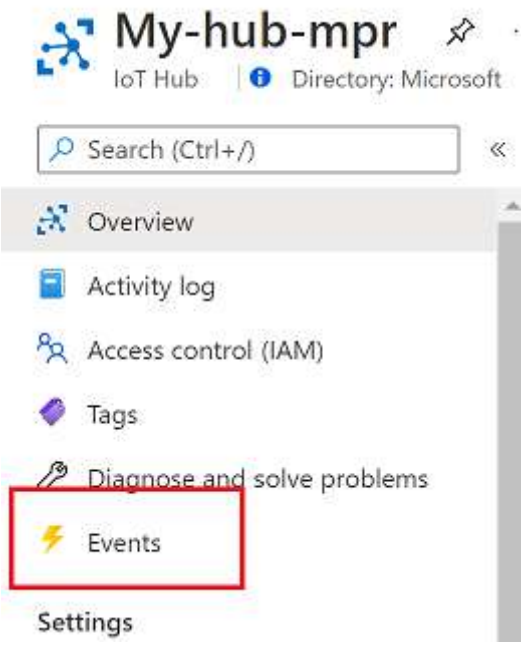
Task 3: Subscribe to the Events

Configure subscription for IoT Hub events

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

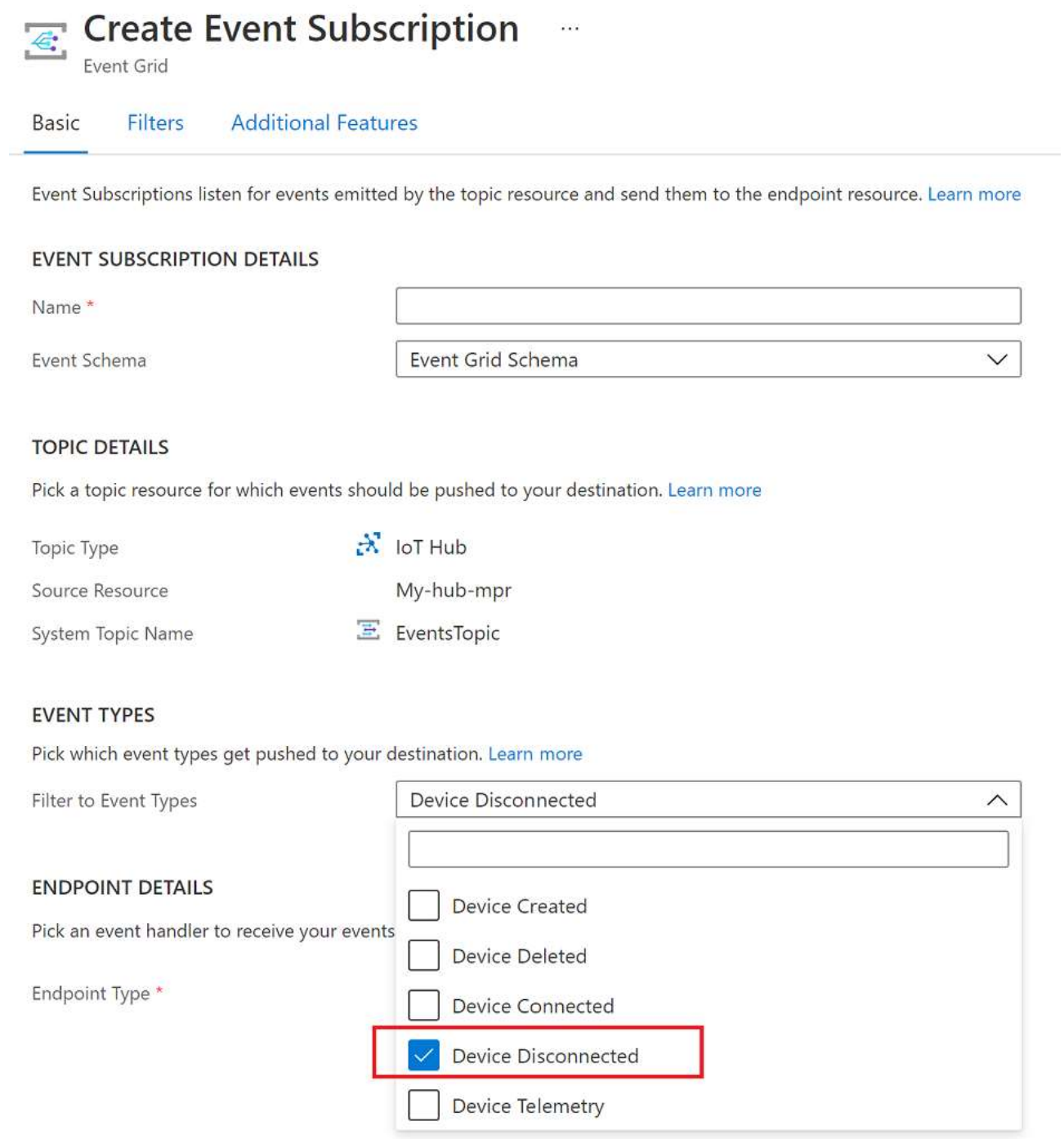
In the Azure portal, navigate to your IoT hub. You can do this by selecting Resource groups, then select the resource group for this tutorial, and then select your IoT hub from the list of resources.

Select Events.



Select + **Event subscription**

Complete the following form:



- **Name**: devicestate
- **Event Schema**: Event Grid Schema
- **Topic Type**: Your Iot Hub created during this training should appears 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**

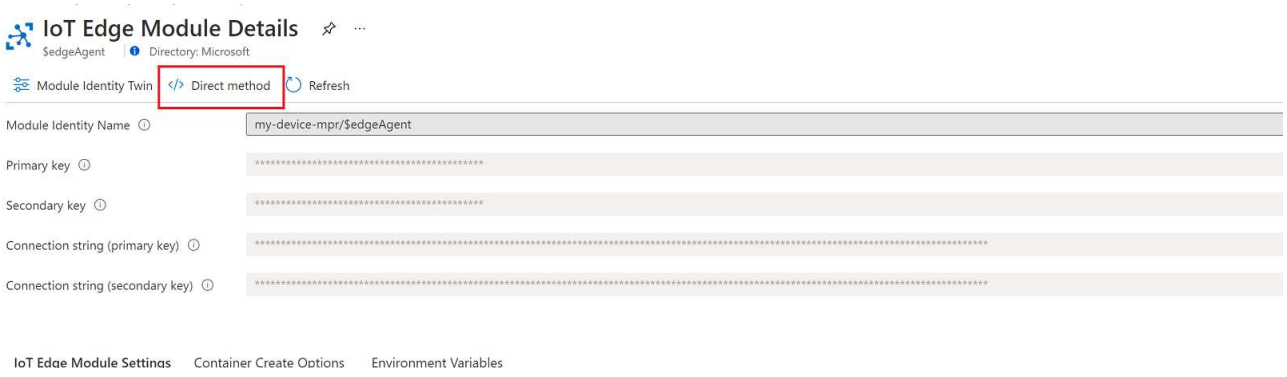
Exercise 6: Interacting with remote devices

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.

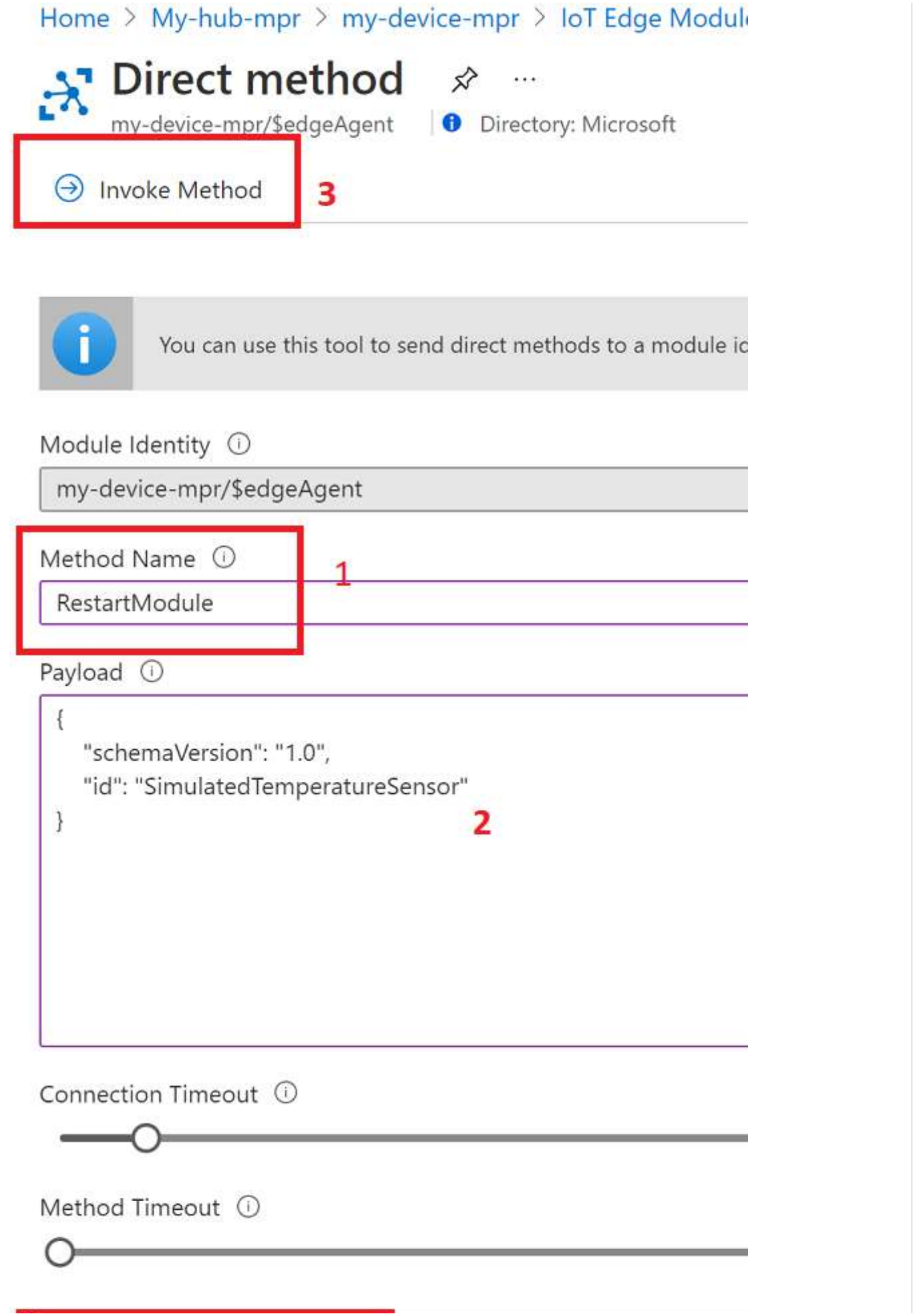
Nex, in your IoT Hub open the Edge device, in **Modules** tab at the bottom look for the **$edgeAgent** Module, click on it **</>Direct Method**



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

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

3. Click on **Invoke Method**


Home > My-hub-mpr > my-device-mpr > IoT Edge Modul
Direct method
my-device-mpr/$edgeAgent
Directory: Microsoft
Invoke Method
3
You can use this tool to send direct methods to a module ic
Module Identity
my-device-mpr/$edgeAgent
Method Name
RestartModule
1
Payload
{
"schemaVersion": "1.0",
"id": "SimulatedTemperatureSensor"
}
2
Connection Timeout
Method Timeout

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

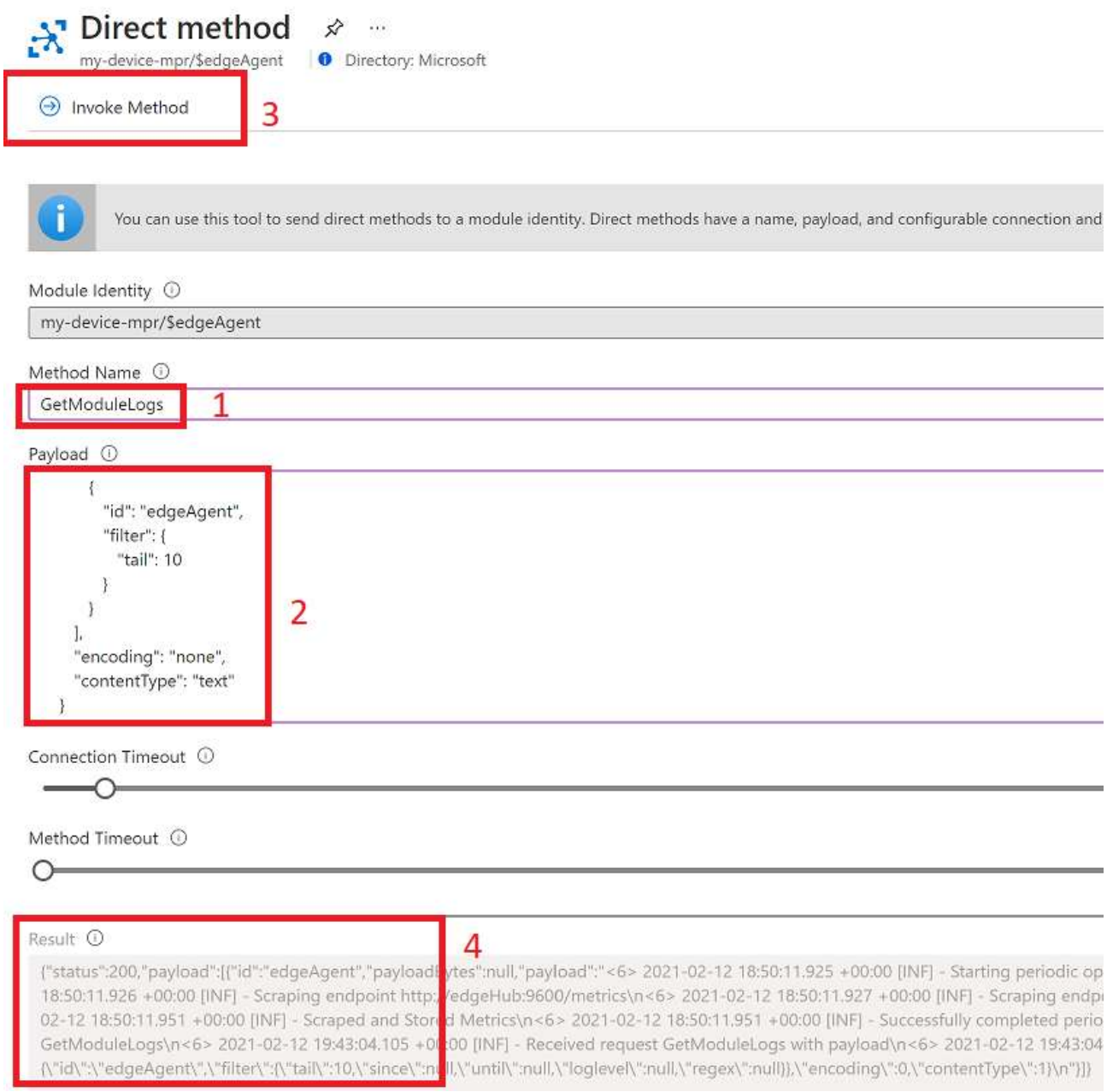
Fill the form again requesting the logs as follow

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.



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 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