Microsoft Cloud Workshop

Internet of Things

Hands-on lab step-by-step

March 2018

Contents

Internet of Things hands-on lab step-by-step

Abstract and learning objectives

This package is designed to guide you through an implementation of an end-to-end IoT solution simulating high velocity data emitted from smart meters and analyzed in Azure. In this session, you will design a lambda architecture, filtering a subset of the telemetry data for real-time visualization on the hot path, and storing all the data in long-term storage for the cold path. After completing the package, you will be better able to implement device registration with the IoT Hub registry and visualize hot data with Power BI.

Learning objectives:

- Implement a simulator sending telemetry from smart meters
- Capture and process both hot and cold data using Stream Analytics and HDInsight with Spark
- Visualize hot data with Power BI

Overview

Fabrikam provides services and smart meters for enterprise energy (electrical power) management. Their "*You-Left-The-Light-On*" service enables the enterprise to understand their energy consumption.

In this hands-on lab, you will construct an end-to-end solution for an IoT scenario that includes device management; telemetry ingest; hot and cold path processing; and reporting.

Requirements

1. Microsoft Azure subscription must be pay-as-you-go or MSDN.
    a. Trial subscriptions will *not* work.
2. A virtual machine configured with:
    a. Visual Studio Community 2017 or later
    b. Azure SDK 2.9 or later (Included with Visual Studio 2017)
3. A running HDInsight Spark cluster (see Before the Hands-on Lab).

Before the hands-on lab

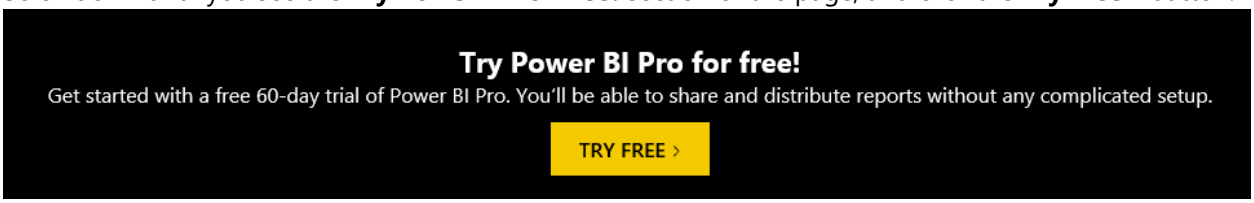
Duration: 45 minutes

In this exercise, you will set up your environment for use in the rest of the hands-on lab. You should follow all the steps provided in the Before the hands-on lab section to prepare your environment *before* attending the hands-on lab.

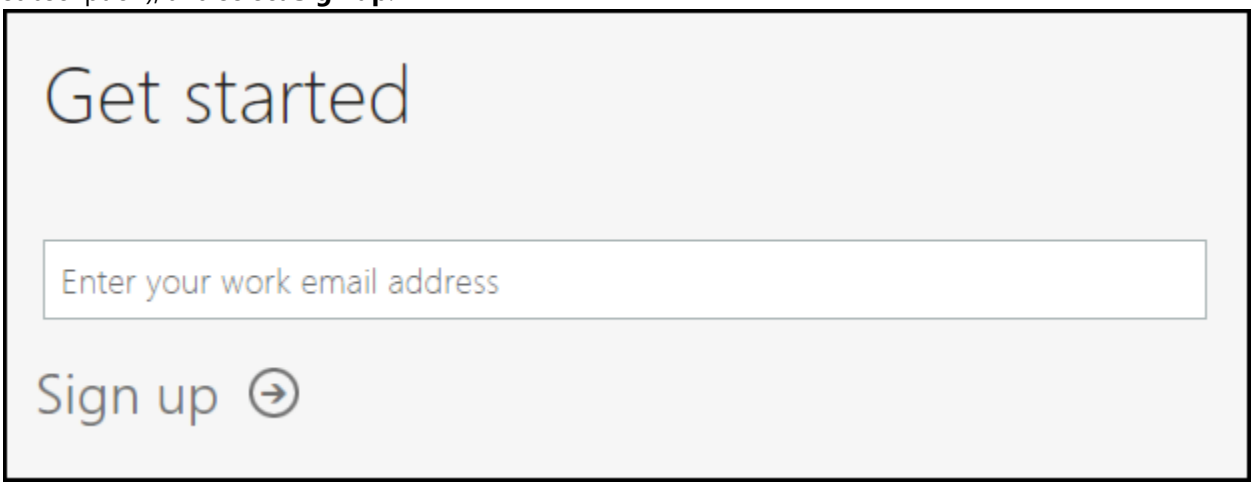
Task 1: Provision Power BI

If you do not already have a Power BI account:

1. Go to https://powerbi.microsoft.com/features/.
2. Scroll down until you see the **Try Power BI for free!** section of the page, and click the **Try Free>** button.



3. On the page, enter your work email address (which should be the same account as the one you use for your Azure subscription), and select **Sign up**.



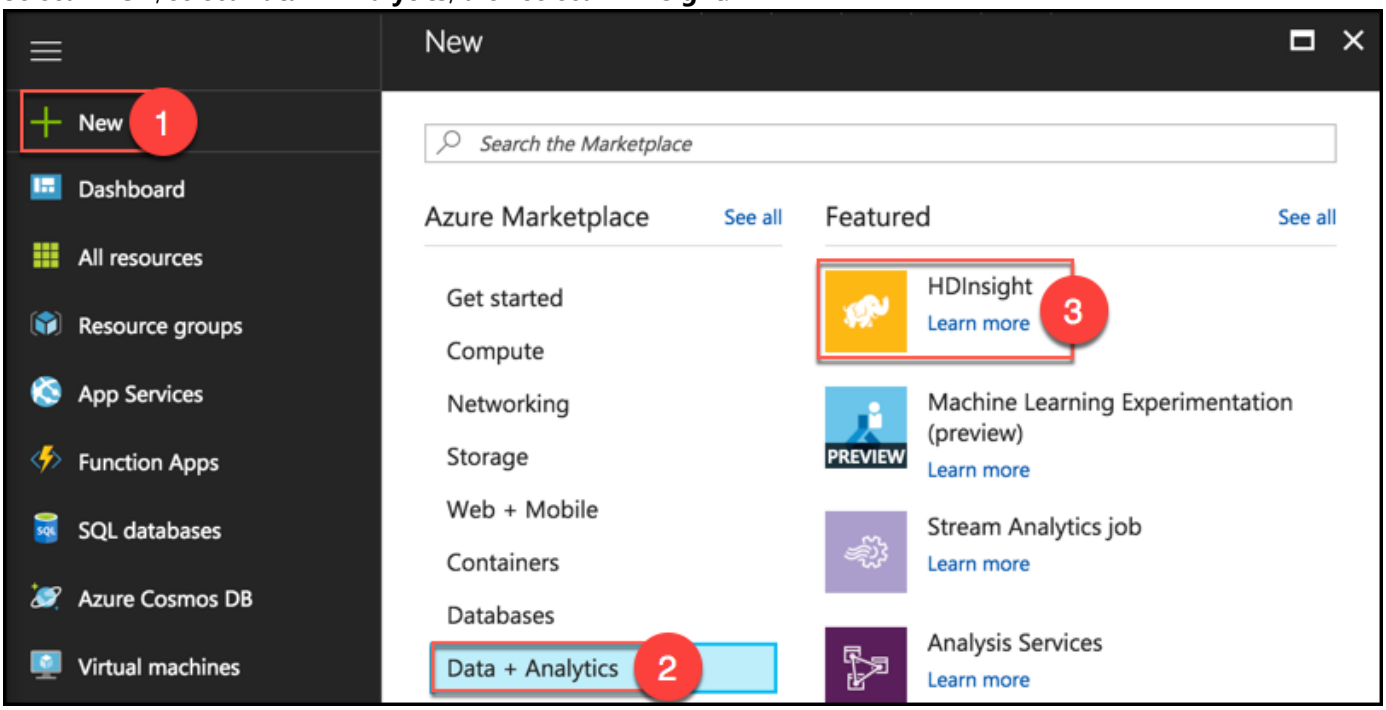
4. Follow the on-screen prompts, and your Power BI environment should be ready within minutes. You can always return to it via https://app.powerbi.com/.

Task 2: Provision an HDInsight with Spark Cluster

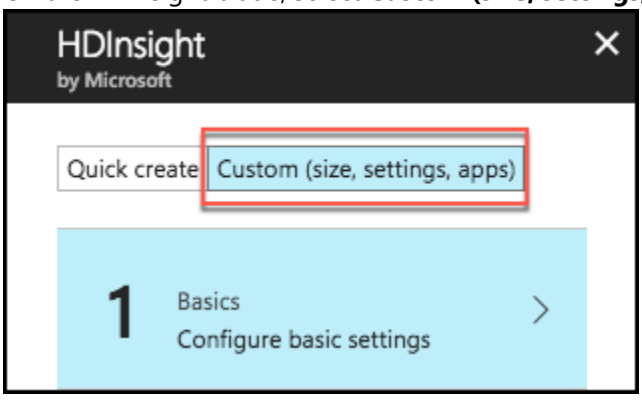
Using the Azure Portal, provision a new HDInsight cluster.

1. Open a browser, and go to the Azure portal (https://portal.azure.com).

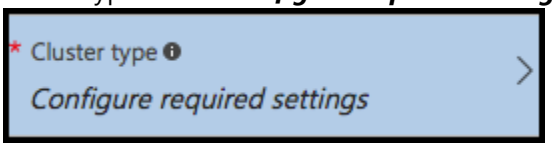
2. Select **+New**, select **Data + Analytics**, then select **HDInsight**.



3. On the HDInsight blade, select **Custom (size, settings, apps)**.

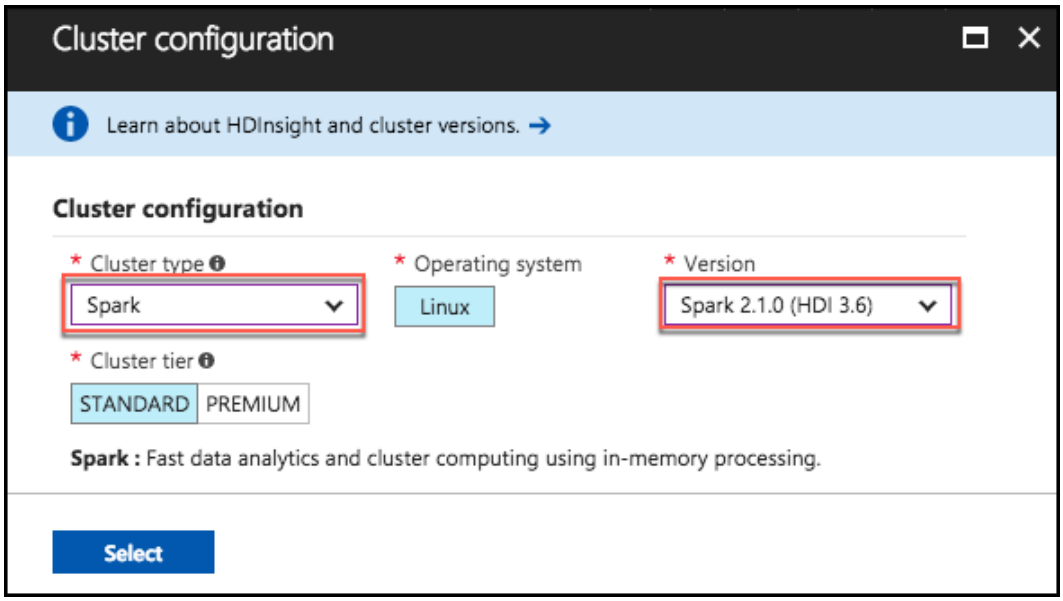


4. On the Basics blade, enter the following settings:
   a. Cluster name: Enter a unique name (verified by the green checkmark).
   b. Subscription: Select the Azure subscription into which you want to deploy the cluster.
   c. Custer type: Select ***Configure required settings***.



      i. On the Cluster configuration blade, set the **Cluster type** to **Spark** and the **Version** to **Spark 2.1.0 (HDI 3.6)**. Note that the Operating System option for the Spark cluster is fixed to Linux.

ii. Cluster tier: Select **Standard**.



iii. Select **Select** to close the Cluster configuration blade.

d. Cluster login username: Leave as **admin**.

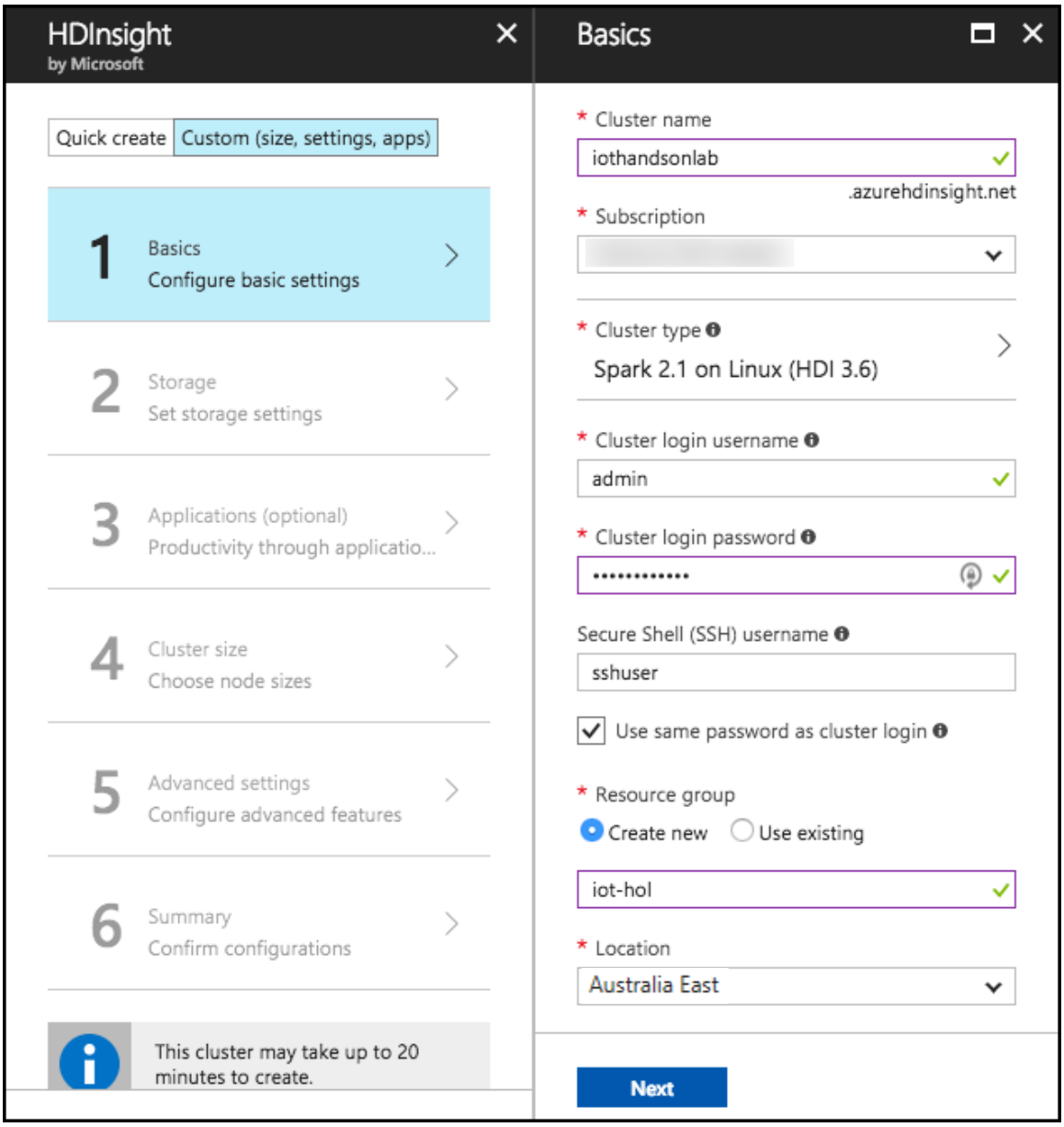
e. Cluster login password: Enter **Password.1!!** for the admin password.

f. Secure Shell (SSH) username: Enter **sshuser**.

g. Use same password as cluster login: Ensure the checkbox is **checked**.

h. Resource group: Select the Create new radio button, and enter **iot-hol** for the resource group name.

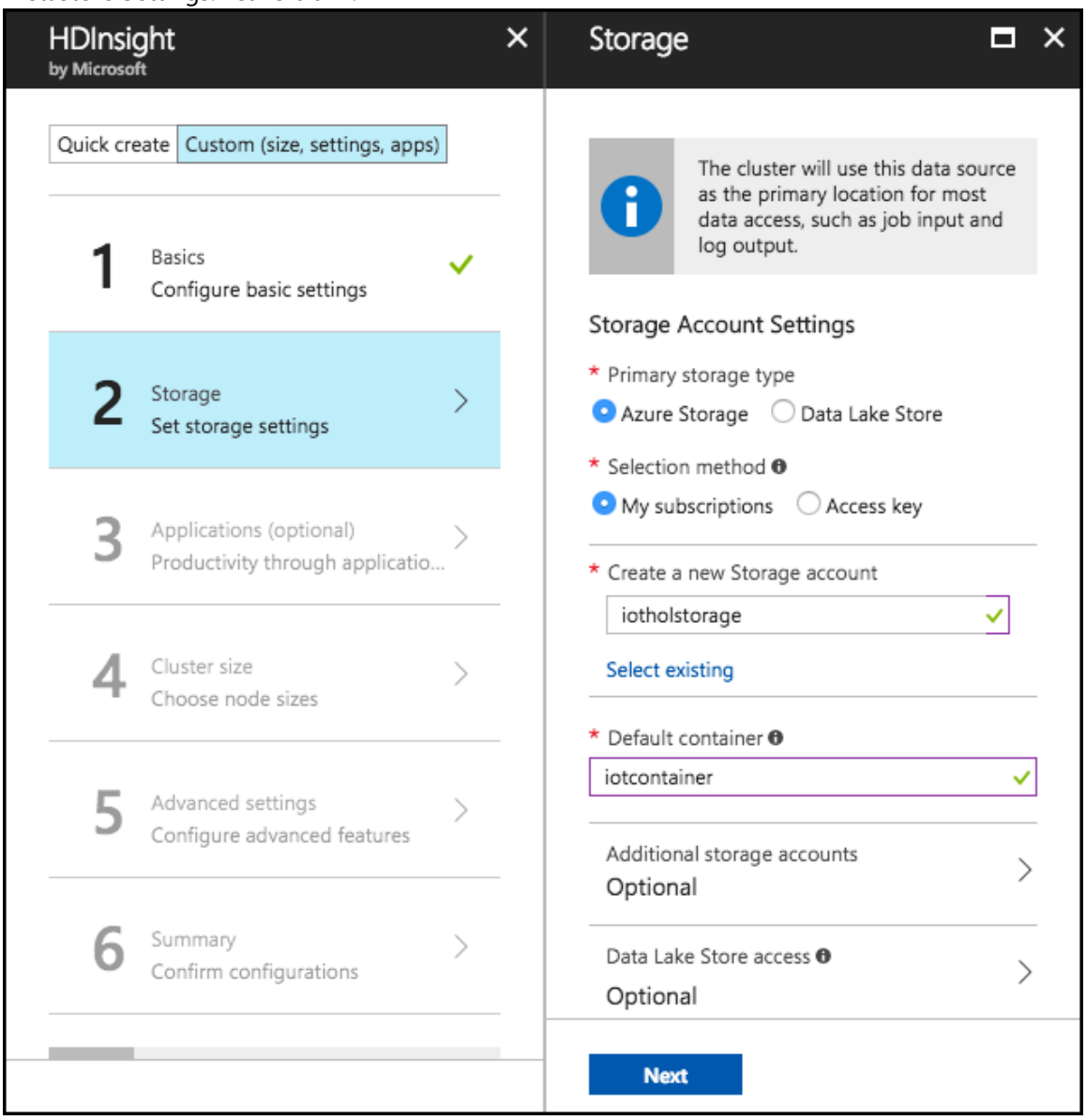
i. Location: Select the desired location (Australia East or Australia Southeast) from the dropdown list, and remember this, as the same location will be used for all other Azure resources.



j. Select **Next** to move on to the storage settings.

5. On the Storage blade:
   a. Primary storage type: Leave set to **Azure Storage.**
   b. Selection Method: Leave set to **My subscriptions**.
   c. Select a Storage account: Select Create new, and enter a name for the storage account, such as iotholstorage.
   d. Default container: Enter **iotcontainer**.
   e. Additional storage accounts: Leave unconfigured.
   f. Data Lake Store access: Leave unconfigured.

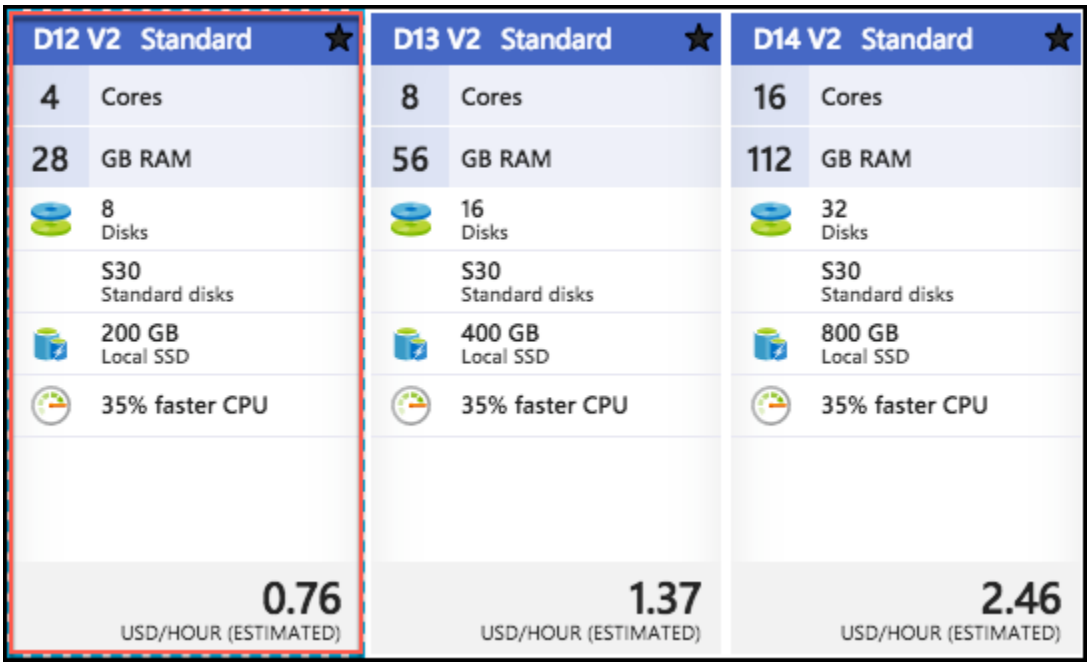
g. Metastore Settings: Leave blank.



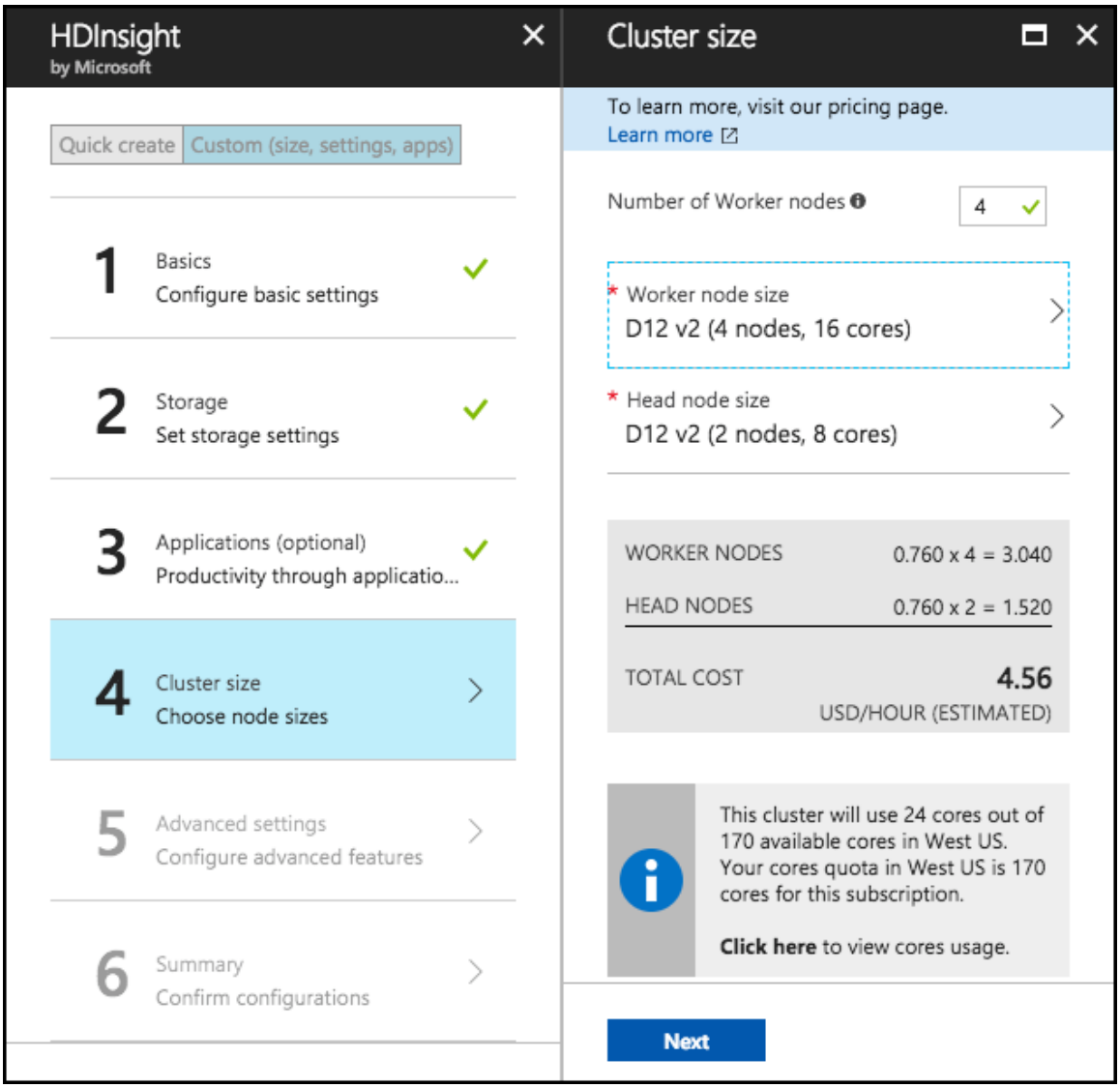
h. Select **Next**.

6. Select **Next** on the Applications (optional) blade. No applications are being added.
7. On the Cluster size blade:
   a. Number of worker nodes: Leave set to **4**.

b. Select **Worker node size**, and select **D12 v2**, then select **Select**.



c. Leave **Head node size**, set to the default, **D12 v2**.

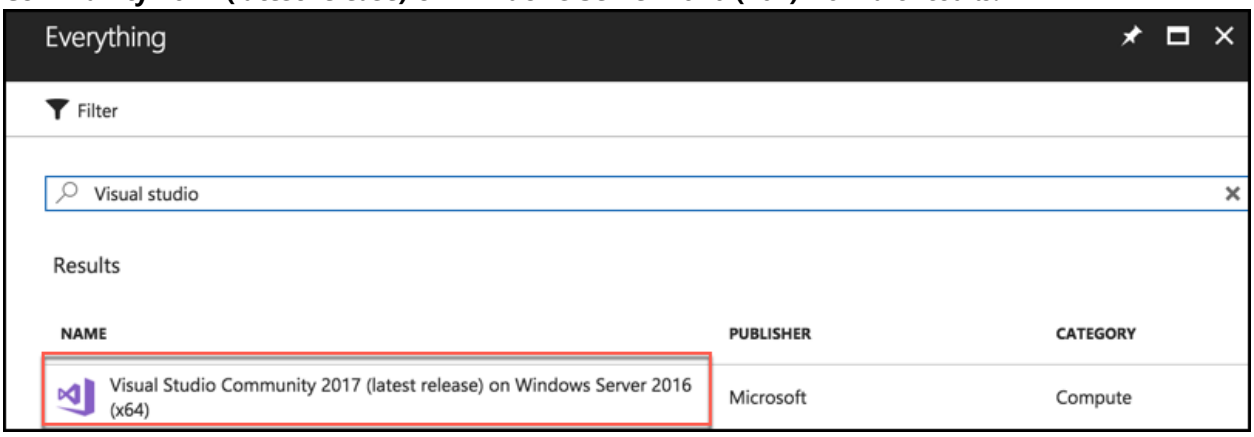


d. Select **Next**.

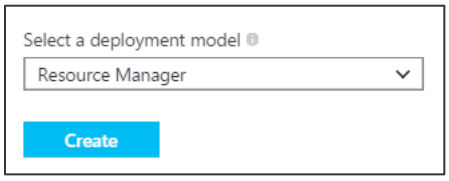
8. Select **Next** on the Advanced settings blade to move to the Cluster summary blade.
9. Select **Create** on the Cluster summary blade to create the cluster.
10. It will take approximately 20 minutes to create your cluster. You can move on to the steps below while the cluster is provisioning.

Task 3: Setup a lab virtual machine (VM) – **Optional**, only if you do not have Visual Studio 2017 installed on your machine

1. In the Azure Portal, select **+New**, then type "Visual Studio" into the search bar. Select **Visual Studio Community 2017 (latest release) on Windows Server 2016 (x64)** from the results**.**

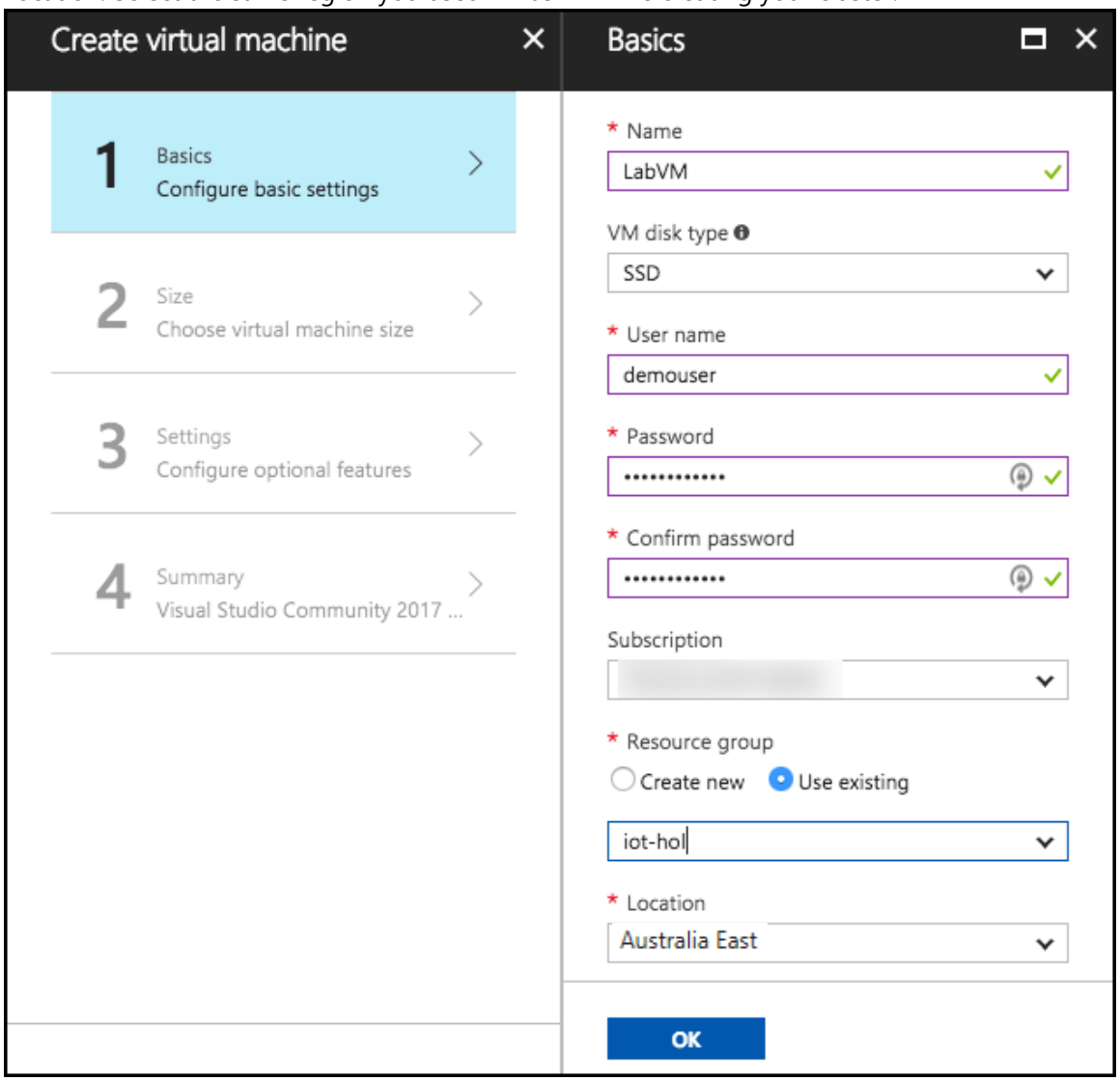


2. On the blade that comes up, at the bottom, ensure the deployment model is set to **Resource Manager** and select **Create**.



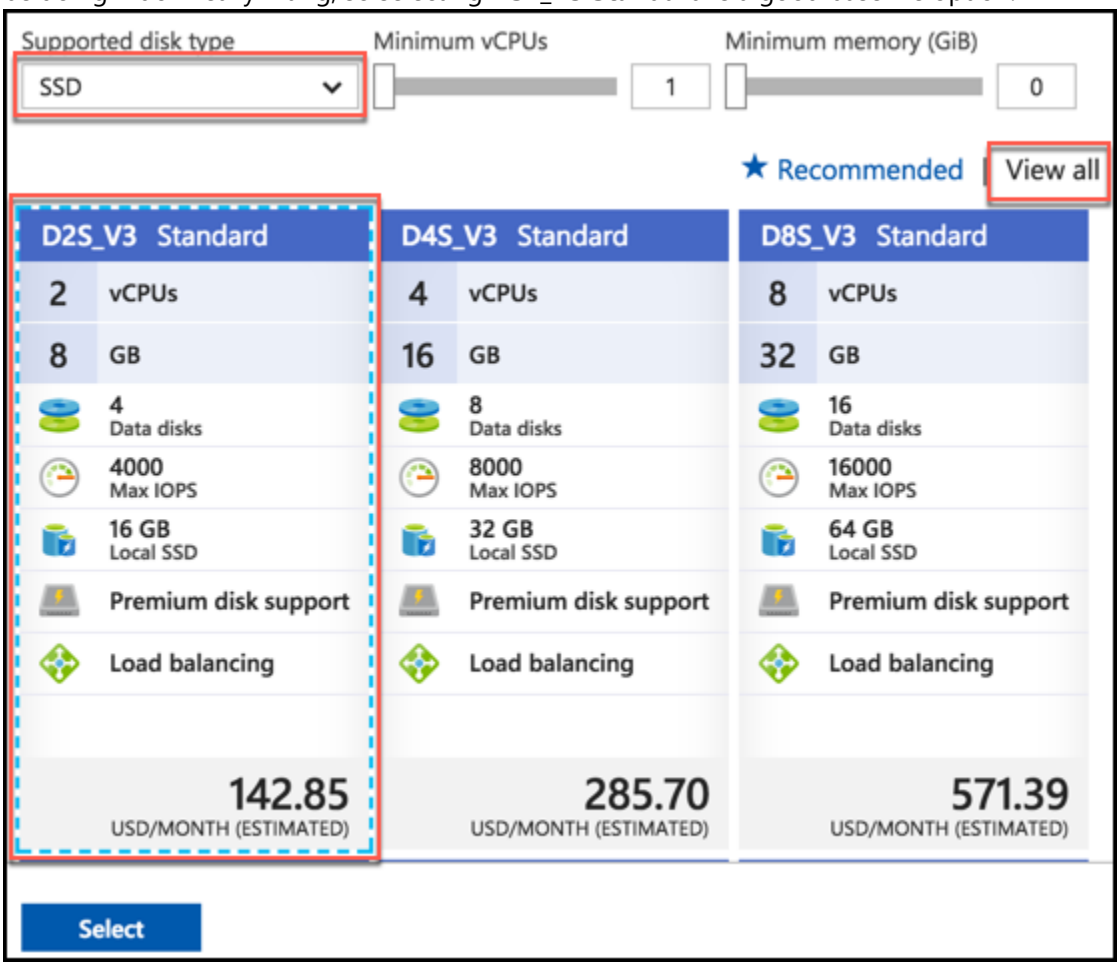
3. Set the following configuration on the Basics tab.
    - Name: Enter **LabVM**.
    - VM disk type: Select **SSD**.
    - User name: Enter **demouser**
    - Password: Enter **Password.1!!**
    - Subscription: Select the same subscription you used to create your cluster in Task 1.
    - Resource Group: Select Use existing, and select the resource group you provisioned while creating your cluster in Task 1.

- Location: Select the same region you used in Task 1 while creating your cluster.



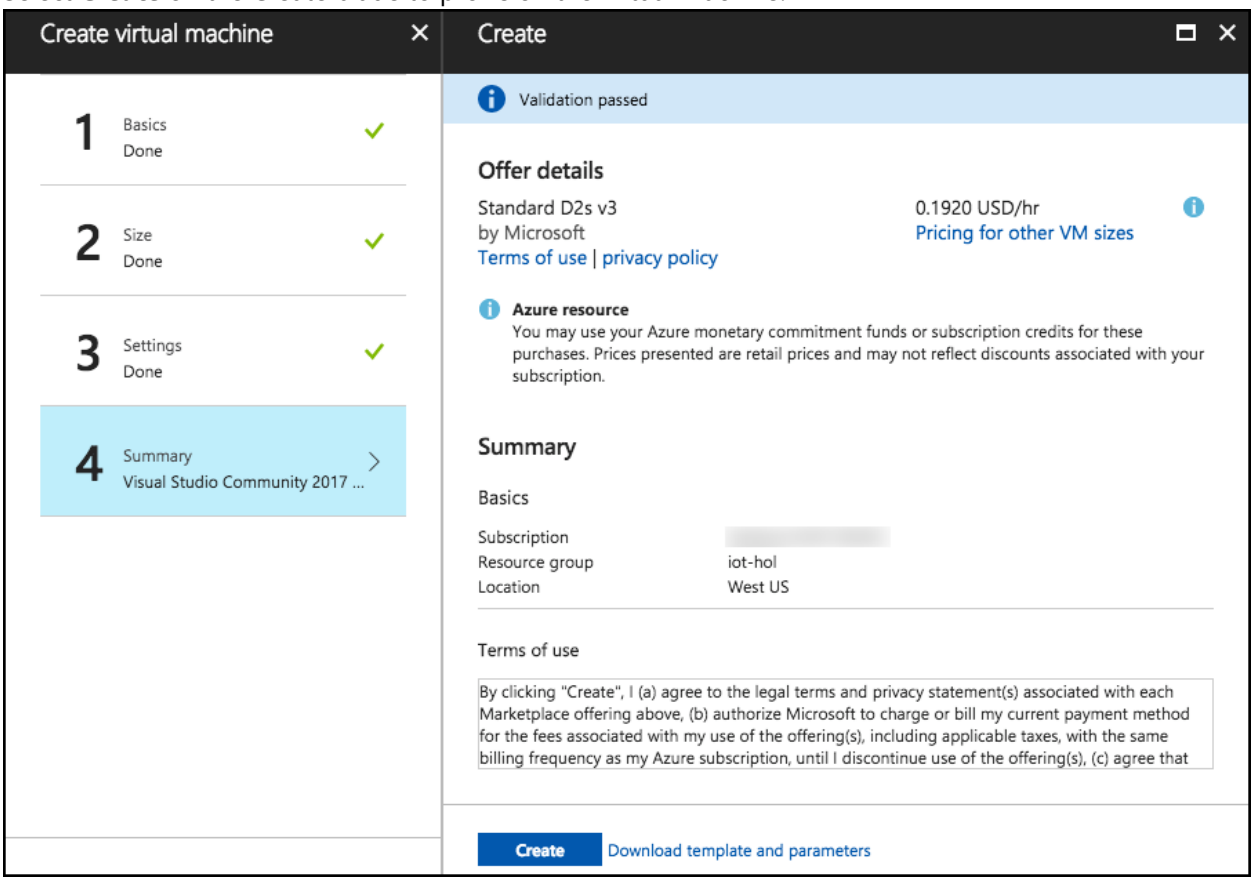
4. Select **OK** to move to the next step.

5. On the Choose a size blade, ensure the Supported disk type is set to SSD, and select View all. This machine won't be doing much heavy lifting, so selecting **DS2_V3 Standard** is a good baseline option.



6. Select **Select** to move on to the Settings blade.
7. Accept all the default values on the Settings blade, and Select **OK**.

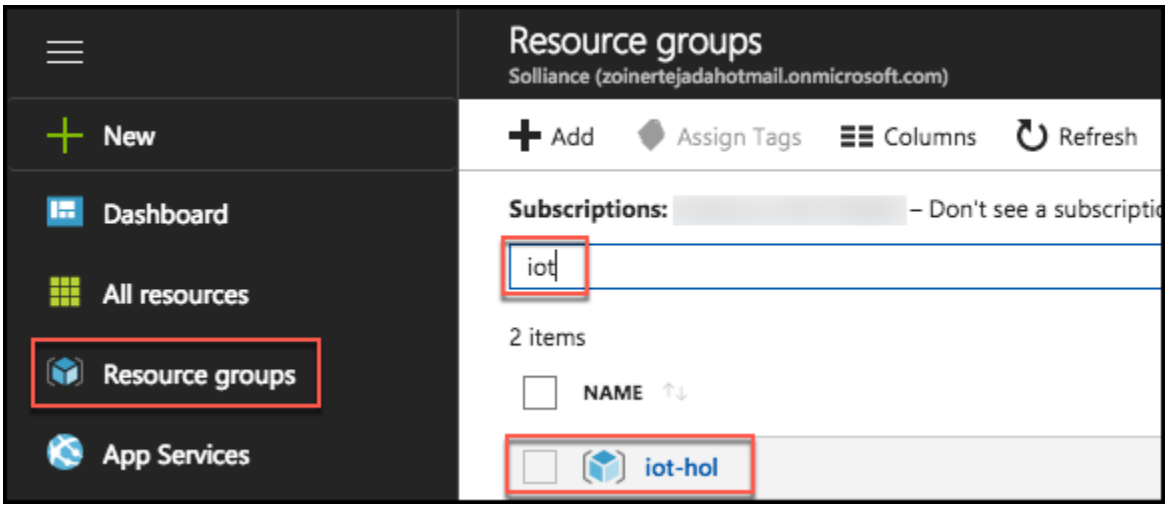
8. Select **Create** on the Create blade to provision the virtual machine.



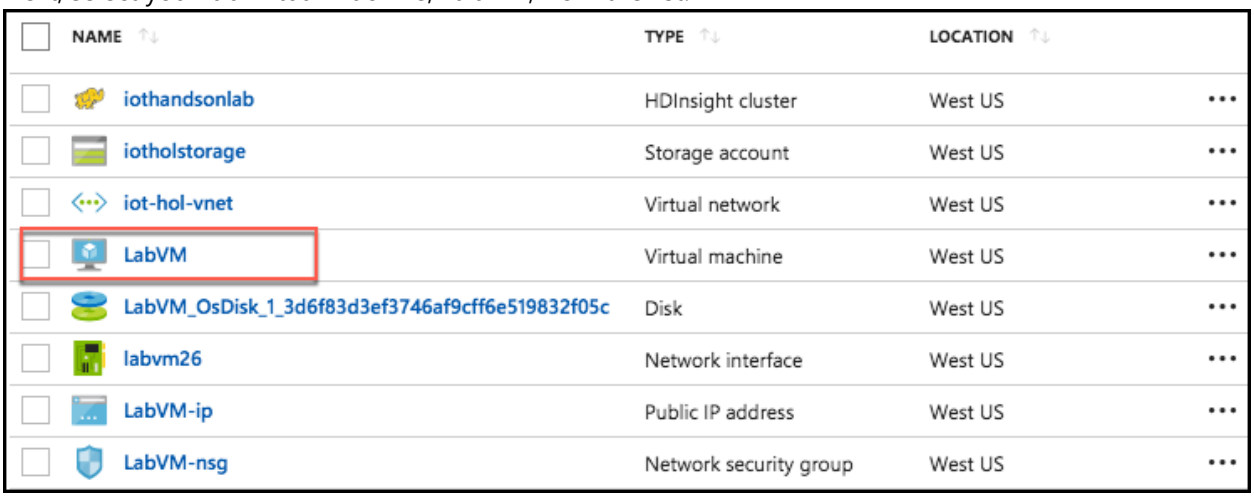
9. It may take 10+ minutes for the virtual machine to complete provisioning.

Task 4: Connect to the lab VM

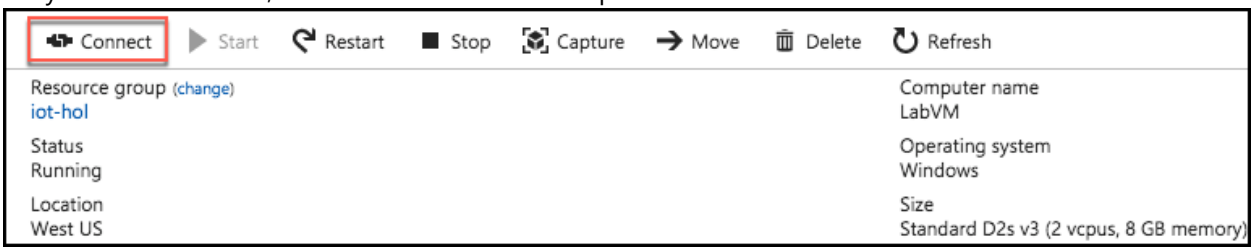
1. Connect to the Lab VM. (If you are already connected to your Lab VM, skip to Step 9.)
2. From the left side menu in the Azure portal, click on **Resource groups**, then enter your resource group name into the filter box, and select it from the list.



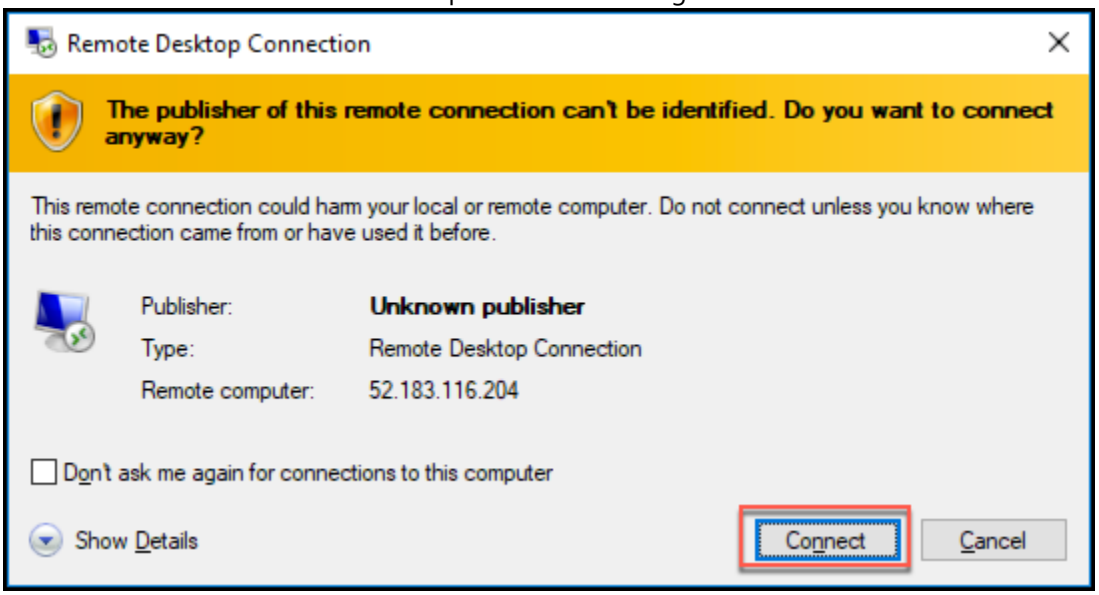
3. Next, select your lab virtual machine, **LabVM**, from the list.



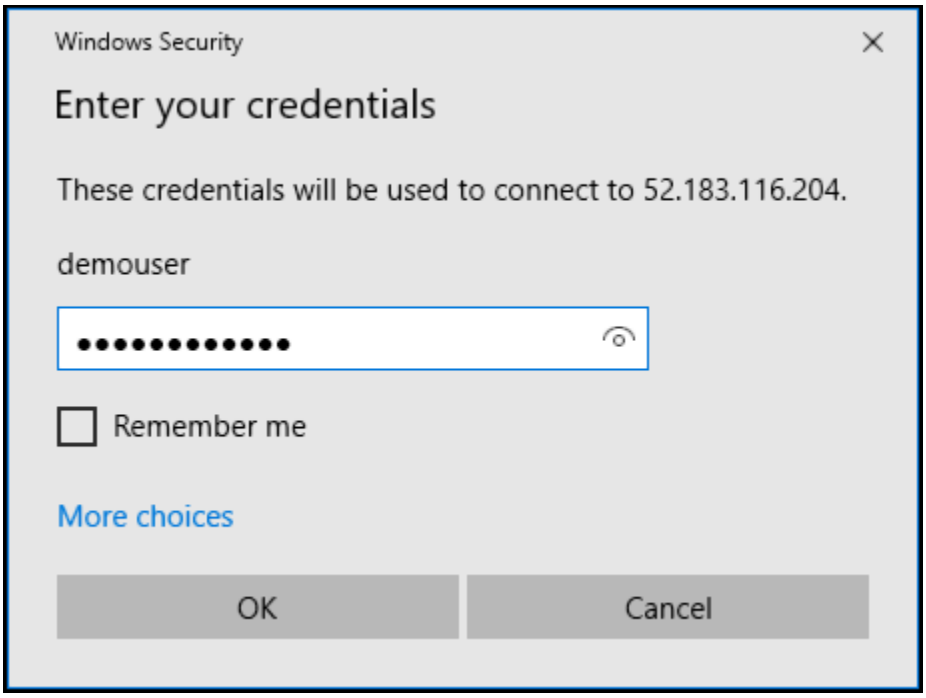
4. On your Lab VM blade, select **Connect** from the top menu.



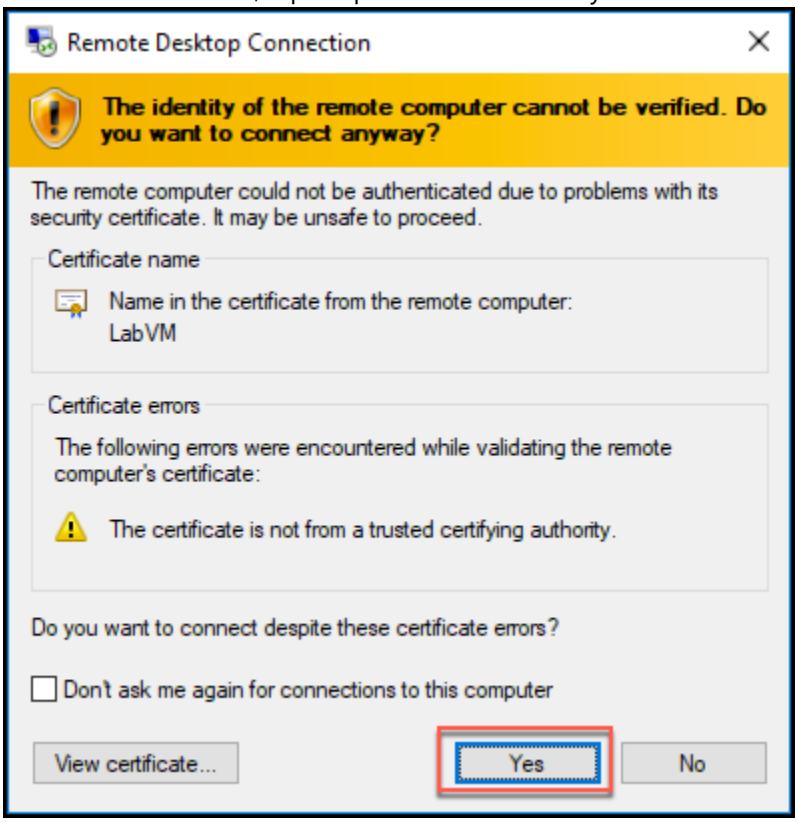
5. Download and open the RDP file.
6. Select Connect on the Remote Desktop Connection dialog.



7. Enter the following credentials (or the non-default credentials if you changed them):
   - User name: **demouser**
   - Password: **Password.1!!**

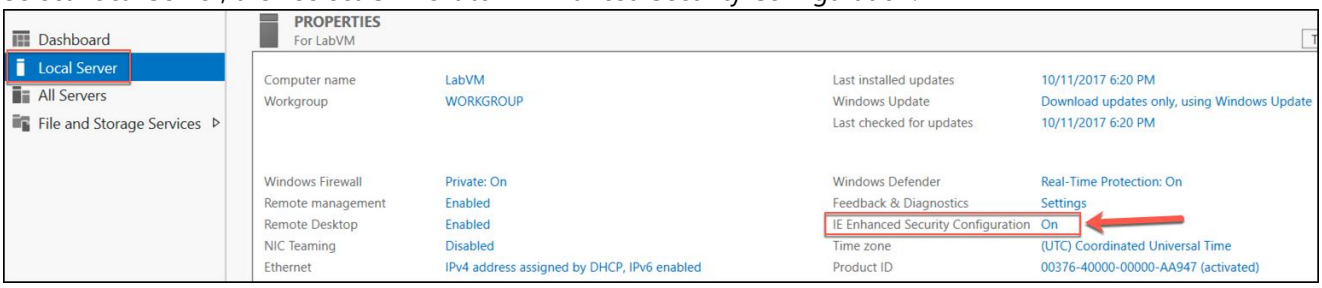


8. Select **Yes** to connect, if prompted that the identity of the remote computer cannot be verified.

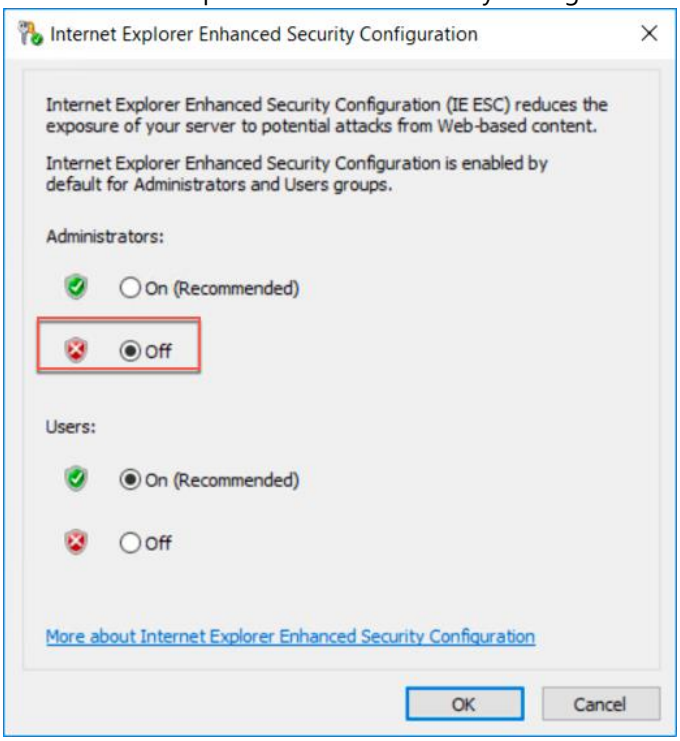


9. Once logged in, launch the Server Manager. This should start automatically, but you can access it via the Start menu if it does not start.

10. Select Local Server, then select On next to IE Enhanced Security Configuration.



11. In the Internet Explorer Enhanced Security Configuration dialog, select Off under Administrators, then select OK.



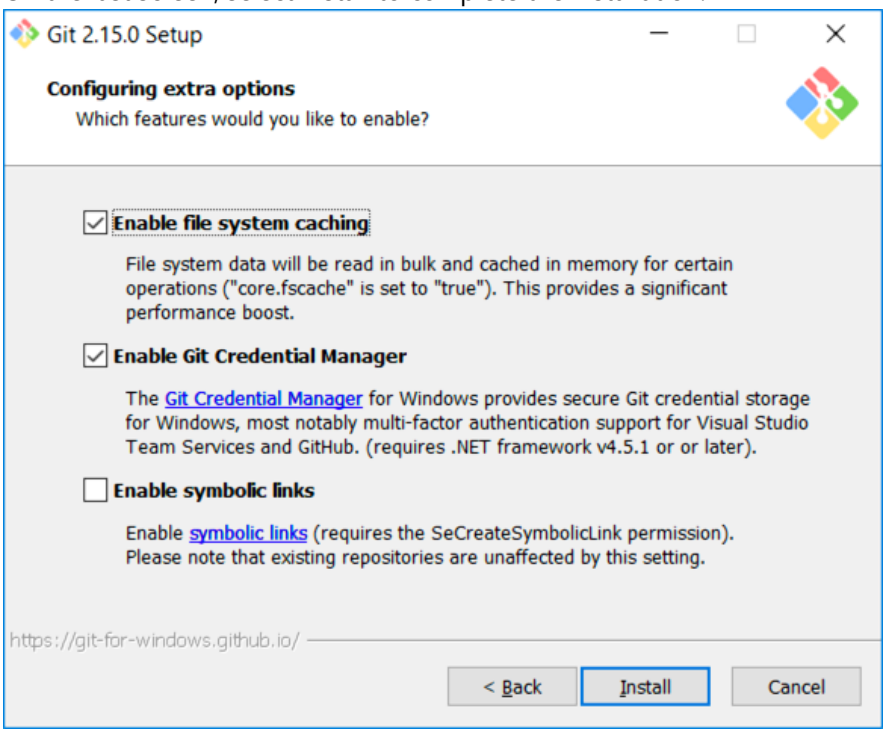
12. Close the Server Manager.

Task 5: Prepare an SSH client

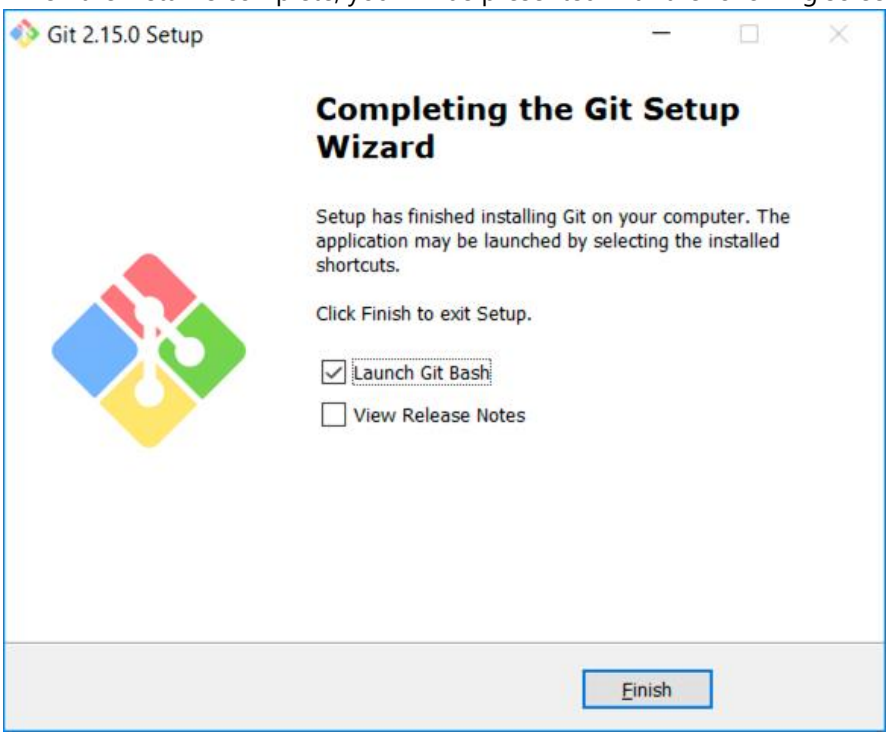
In this task, you will download, install, and prepare the Git Bash SSH client that you will use to access your HDInsight cluster from your Lab VM.

1. On your Lab VM, open a browser, and navigate to https://git-scm.com/downloads to download Git Bash.

2. Select the download for your OS, and then select the Download 2.16.x for... button.
3. Run the downloaded installer, select Next on each screen to accept the defaults.
4. On the last screen, select Install to complete the installation.



5. When the install is complete, you will be presented with the following screen:



6. Check the Launch Git Bash checkbox, and uncheck View Release Notes. Select Finish.
7. Leave the bash window open, as you will use it later in this lab.

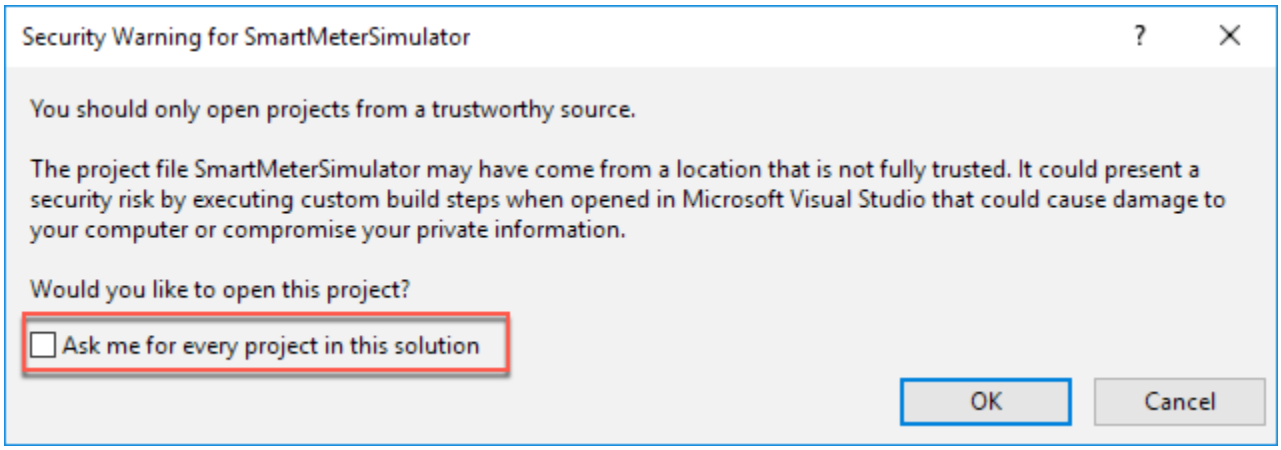
Exercise 1: Environment setup

Duration: 10 minutes

Fabrikam has provided a Smart Meter Simulator that they use to simulate device registration, as well as the generation and transmission of telemetry data. They have asked you to use this as the starting point for integrating their smart meters with Azure.

Task 1: Download and open the Smart Meter Simulator project

1. Connect to your Lab VM, as was detailed in Before the Hands-on Lab, Task 4.
2. From your Lab VM, download the Smart Meter Simulator starter project from the following URL: https://bit.ly/2wMSwsH (Note: the URL is case-sensitive)
3. Unzip the contents to the folder C:\SmartMeter\.
4. Open SmartMeterSimulator.sln with Visual Studio 2017.
5. Sign in to Visual Studio or create an account, if prompted.
6. If the Security Warning for SmartMeterSimulator window appears, uncheck *Ask me for every project in this solution*, and select OK.



Note: If you attempt to build the solution at this point, you will see many build errors. This is intentional. You will correct these in the exercises that follow.

Exercise 2: IoT Hub provisioning

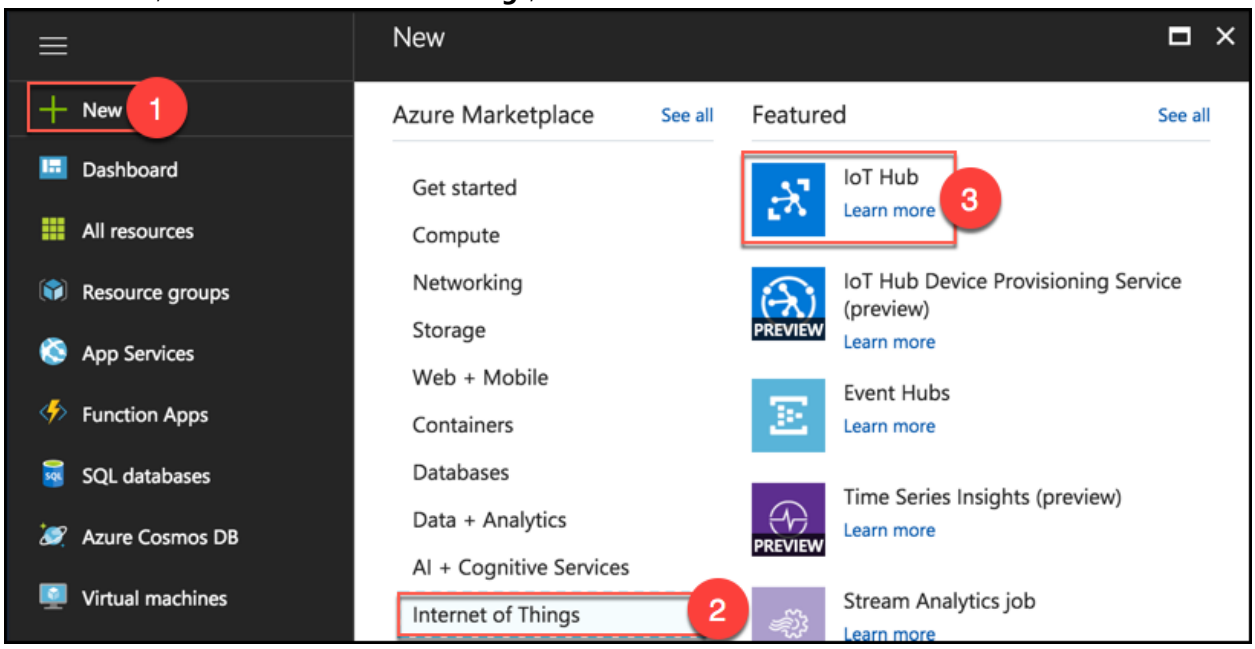
Duration: 20 minutes

In your architecture design session with Fabrikam, it was agreed that you would use an Azure IoT Hub to manage both the device registration and telemetry ingest from the Smart Meter Simulator. Your team also identified the Microsoft provided Device Explorer project that Fabrikam can use to view the list and status of devices in the IoT Hub registry.

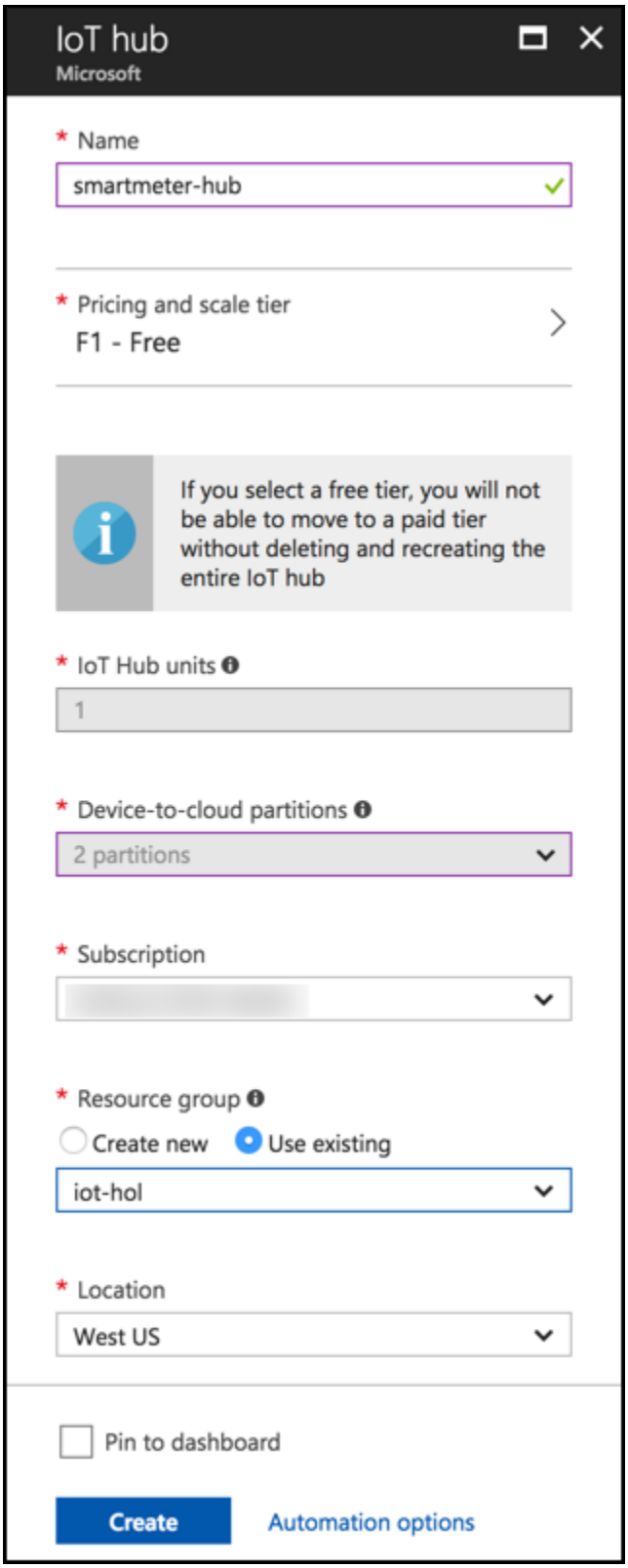
Task 1: Provision an IoT Hub

In these steps, you will provision an instance of IoT Hub.

1. In a browser, navigate to the Azure Portal (https://portal.azure.com).
2. Select **+New**, then select **Internet of Things**, and select **IoT Hub**.

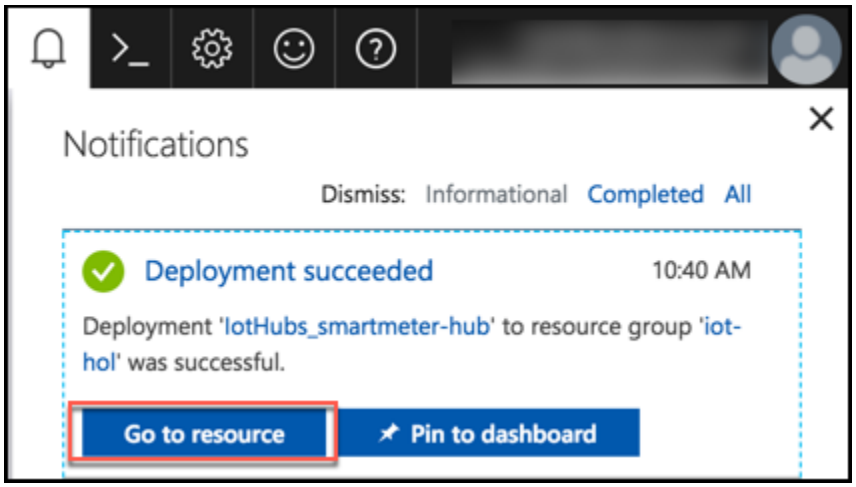


3. In the **IoT Hub** blade, enter the following:
    a. Name: Provide a name for your new IoT Hub, such as smartmeter-hub
    b. Pricing and scale tier: Select **F1 Free**
    c. IoT Hub units: Set to **1** automatically when the F1 pricing tier is selected.
    d. Device-to-cloud partitions: Set to **2 partitions** automatically when the F1 pricing tier is selected.
    e. Subscription: Select the same subscription you've been using for previous resources in this lab.
    f. Resource group: Select Use existing, and select the **iot-hol** resource group you created previously.
    g. Location: Select the location you used previously.

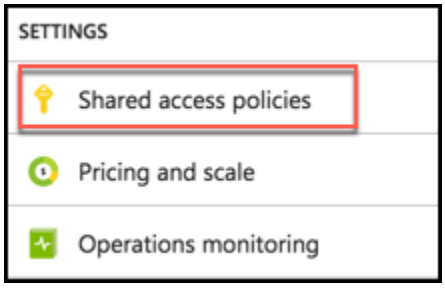


h. Select **Create**.

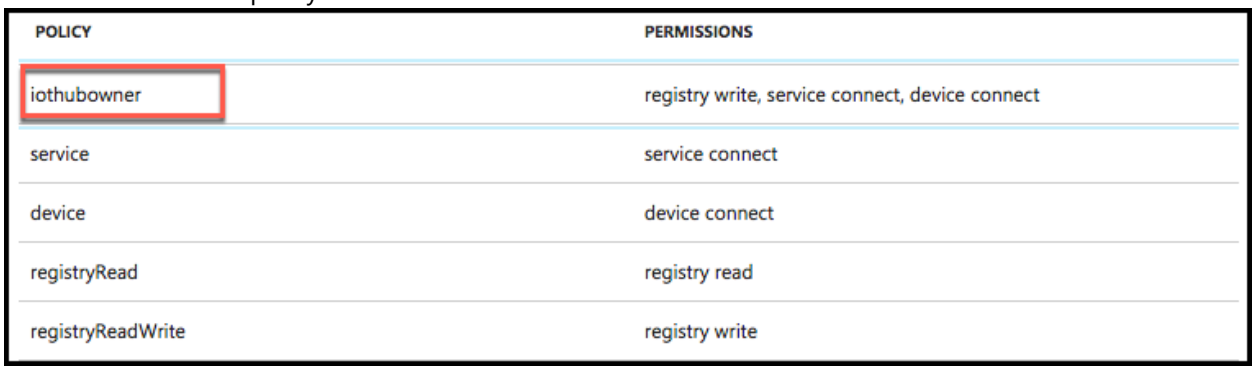
4. When the IoT Hub deployment is completed, you will receive a notification in the Azure portal. Navigate to your new IoT Hub by selecting Go to resource in the notification.



5. From the IoT Hub's Overview blade, select Shared access policies under Settings on the left-hand menu.

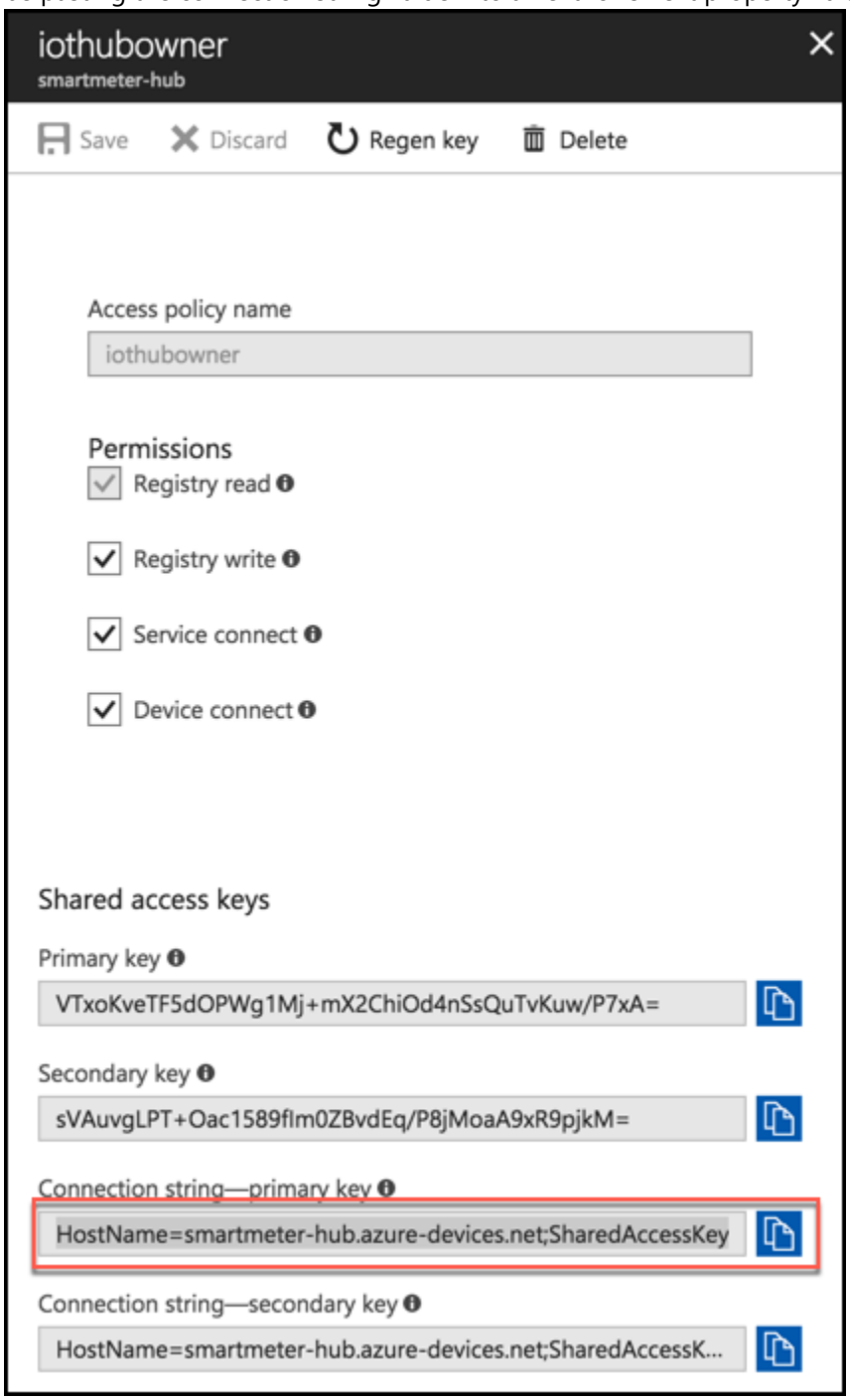


6. Select **iothubowner** policy.



POLICY	PERMISSIONS
iothubowner	registry write, service connect, device connect
service	service connect
device	device connect
registryRead	registry read
registryReadWrite	registry write

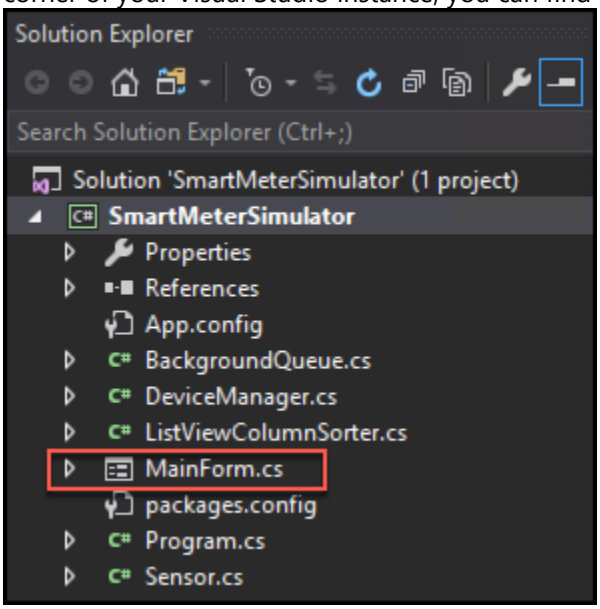
7. In the iothubowner blade, select the **Copy** button to the right of the **Connection string - primary key** field. You will be pasting the connection string value into a TextBox's Text property value in the next Task.



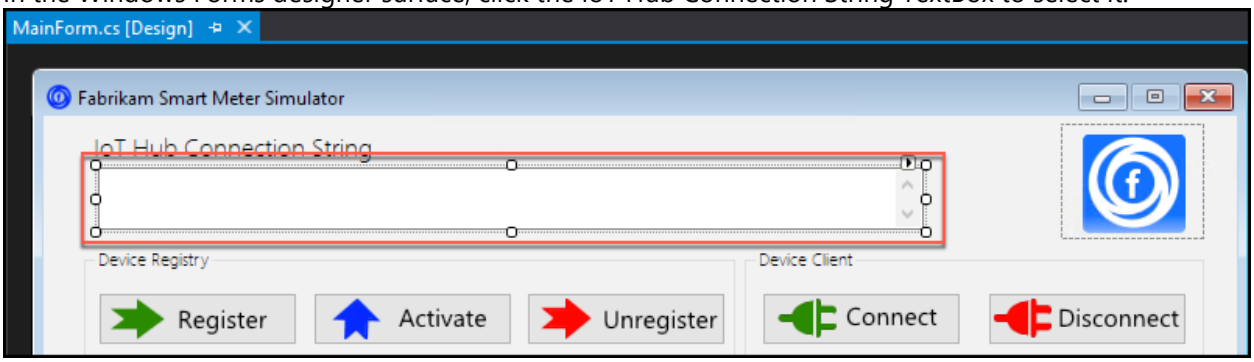
Task 2: Configure the Smart Meter Simulator

If you want to save this connection string with your project (in case you stop debugging or otherwise close the simulator), you can set this as the default text for the text box. Follow these steps to configure the connection string:

1. Return to Visual Studio on your Lab VM.
2. In the Solution Explorer, double-click **MainForm.cs** to open it. (If the Solution Explorer is not in the upper left corner of your Visual Studio instance, you can find it under the View menu in Visual Studio.)

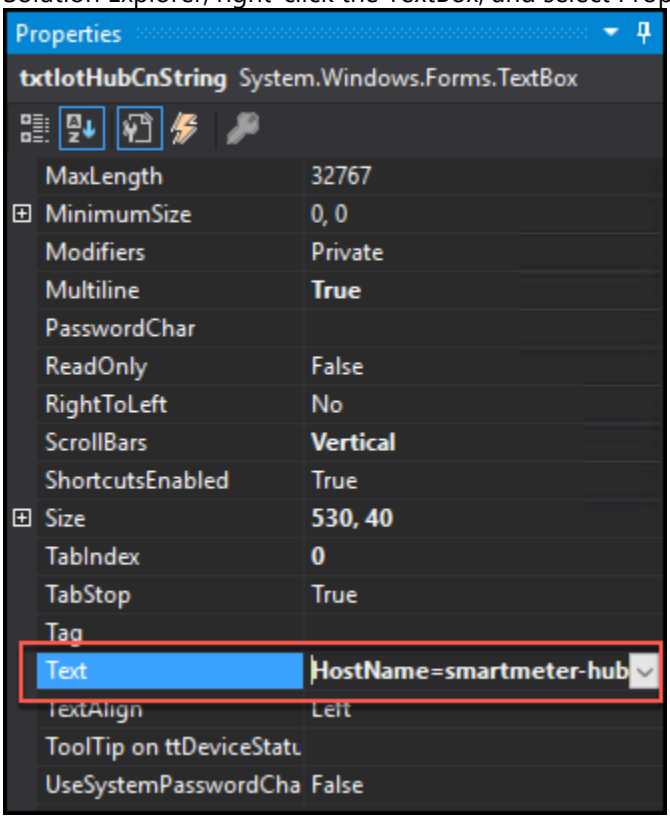


3. In the Windows Forms designer surface, click the IoT Hub Connection String TextBox to select it.

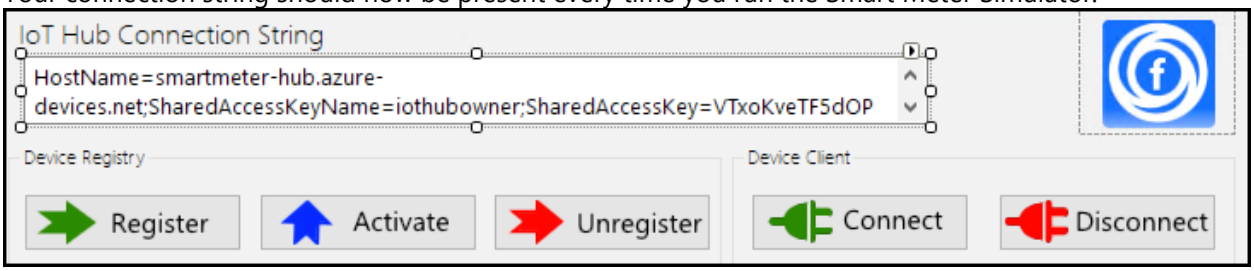


4. In the Properties panel, scroll until you see the **Text** property. Paste your IoT Hub connection string value copied in step 7 of the previous task into the value for the Text property. (If the properties window is not visible below the

Solution Explorer, right-click the TextBox, and select Properties.)



5. Your connection string should now be present every time you run the Smart Meter Simulator.



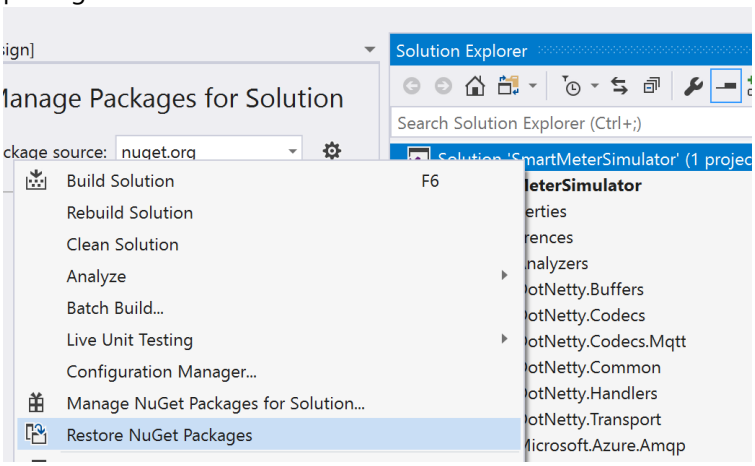
Exercise 3: Completing the Smart Meter Simulator

Duration: 60 minutes

Fabrikam has left you a partially completed sample in the form of the Smart Meter Simulator solution. You will need to complete the missing lines of code that deal with device registration management and device telemetry transmission that communicate with your IoT Hub.

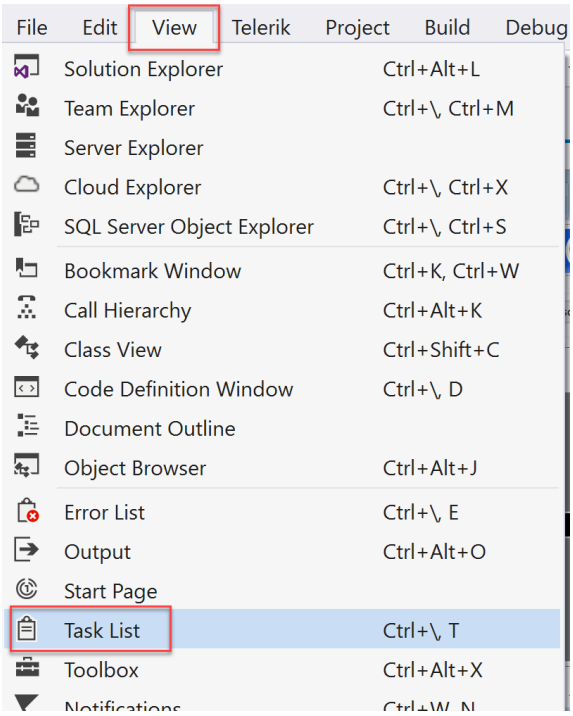
Task 0: Restore NuGet packages

1. In Visual Studio on your Lab VM, in the Solution Explorer, right-mouse click on **Solution 'SmartMeterSimulator'** and select **Restore NuGet Packages**. You may have to update the packages and accept the license terms of the packages.



Task 1: Implement device management with the IoT Hub

2. In Visual Studio on your Lab VM, use Solution Explorer to open the file **DeviceManager.cs**.
3. From the Visual Studio **View** menu, click **Task List**.



4. There you will see a list of TODO tasks, where each task represents one line of code that needs to be completed. Copy the code in **bold** below the corresponding TODO item in the completed code that follows.
5. The following code represents the completed tasks in DeviceManager.cs:

```
class DeviceManager
{
    static string connectionString;
    static RegistryManager registryManager;

    public static string HostName { get; set; }

    public static void IotHubConnect(string cnString)
    {
        connectionString = cnString;

        //TODO: 1.Create an instance of RegistryManager from connectionString
        registryManager = RegistryManager.CreateFromConnectionString(connectionString);

        var builder = IotHubConnectionStringBuilder.Create(cnString);

        HostName = builder.HostName;
    }

    /// <summary>
    /// Register a single device with the IoT hub. The device is initially registered in a
    /// disabled state.
    /// </summary>
    /// <param name="connectionString"></param>
    /// <param name="deviceId"></param>
    /// <returns></returns>
```

```
    public async static Task<string> RegisterDevicesAsync(string connectionString, string
deviceId)
    {
        //Make sure we're connected
        if (registryManager == null)
            IotHubConnect(connectionString);

        //TODO: 2.Create new device
        Device device = new Device(deviceId);

        //TODO: 3.Initialize device with a status of Disabled
        //Enabled in a subsequent step
        device.Status = DeviceStatus.Disabled;

        try
        {
            //TODO: 4.Register the new device
            device = await registryManager.AddDeviceAsync(device);
        }
        catch (Exception ex)
        {
            if (ex is DeviceAlreadyExistsException ||
                ex.Message.Contains("DeviceAlreadyExists"))
            {
                //TODO: 5.Device already exists, get the registered device
                device = await registryManager.GetDeviceAsync(deviceId);

                //TODO: 6.Ensure the device is disabled until Activated later
                device.Status = DeviceStatus.Disabled;

                //TODO: 7.Update IoT Hubs with the device status change
                await registryManager.UpdateDeviceAsync(device);
            }
            else
            {
                MessageBox.Show($"An error occurred while registering one or more
devices:\r\n{ex.Message}");
            }
        }

        //return the device key
        return device.Authentication.SymmetricKey.PrimaryKey;
    }

    /// <summary>
    /// Activate an already registered device by changing its status to Enabled.
    /// </summary>
    /// <param name="connectionString"></param>
    /// <param name="deviceId"></param>
    /// <param name="deviceKey"></param>
    /// <returns></returns>
```

```
    public async static Task<bool> ActivateDeviceAsync(string connectionString, string
deviceId, string deviceKey)
    {
        //Server-side management function to enable the provisioned device
        //to connect to IoT Hub after it has been installed locally.
        //If device id device key are valid, Activate (enable) the device.

        //Make sure we're connected
        if (registryManager == null)
            IotHubConnect(connectionString);

        bool success = false;
        Device device;

        try
        {
            //TODO: 8.Fetch the device
            device = await registryManager.GetDeviceAsync(deviceId);

            //TODO: 9.Verify the device keys match
            if (deviceKey == device.Authentication.SymmetricKey.PrimaryKey)
            {
                //TODO: 10.Enable the device
                device.Status = DeviceStatus.Enabled;

                //TODO: 11.Update IoT Hubs
                await registryManager.UpdateDeviceAsync(device);

                success = true;
            }
        }
        catch(Exception)
        {
            success = false;
        }

        return success;
    }

    /// <summary>
    /// Deactivate a single device in the IoT Hub registry.
    /// </summary>
    /// <param name="connectionString"></param>
    /// <param name="deviceId"></param>
    /// <returns></returns>
    public async static Task<bool> DeactivateDeviceAsync(string connectionString, string
deviceId)
    {
        //Make sure we're connected
        if (registryManager == null)
            IotHubConnect(connectionString);
```

```
        bool success = false;
        Device device;

        try
        {
            //TODO: 12.Lookup the device from the registry by deviceId
            device = await registryManager.GetDeviceAsync(deviceId);

            //TODO: 13.Disable the device
            device.Status = DeviceStatus.Disabled;

            //TODO: 14.Update the registry
            await registryManager.UpdateDeviceAsync(device);

            success = true;
        }
        catch (Exception)
        {
            success = false;
        }

        return success;
    }

    /// <summary>
    /// Unregister a single device from the IoT Hub Registry
    /// </summary>
    /// <param name="connectionString"></param>
    /// <param name="deviceId"></param>
    /// <returns></returns>
    public async static Task UnregisterDevicesAsync(string connectionString, string
deviceId)
    {
        //Make sure we're connected
        if (registryManager == null)
            IotHubConnect(connectionString);

            //TODO: 15.Remove the device from the Registry
            await registryManager.RemoveDeviceAsync(deviceId);
    }

    /// <summary>
    /// Unregister all the devices managed by the Smart Meter Simulator
    /// </summary>
    /// <param name="connectionString"></param>
    /// <returns></returns>
    public async static Task UnregisterAllDevicesAsync(string connectionString)
    {
        //Make sure we're connected
        if (registryManager == null)
           IotHubConnect(connectionString);
```



```
        for(int i = 0; i <= 9; i++)
        {
            string deviceId = "Device" + i.ToString();

            //TODO: 16.Remove the device from the Registry
            await registryManager.RemoveDeviceAsync(deviceId);
        }
    }
}
```

6. Save DeviceManager.cs.

Task 2: Implement the communication of telemetry with the IoT Hub

1. Open Sensor.cs from the Solution Explorer, and complete the TODO items indicated within the code that are responsible for transmitting telemetry data to the IoT Hub.
2. The following code shows the completed result:

```
class Sensor
{
    private DeviceClient _DeviceClient;
    private string _IotHubUri { get; set; }
    public string DeviceId { get; set; }
    public string DeviceKey { get; set; }
    public DeviceState State { get; set; }
    public string StatusWindow { get; set; }
    public double CurrentTemperature
    {
        get
        {
            double avgTemperature = 70;
            Random rand = new Random();

            double currentTemperature = avgTemperature + rand.Next(-6, 6);

            if(currentTemperature <= 68)
                TemperatureIndicator = SensorState.Cold;
            else if(currentTemperature > 68 && currentTemperature < 72)
                TemperatureIndicator = SensorState.Normal;
            else if(currentTemperature >= 72)
                TemperatureIndicator = SensorState.Hot;

            return currentTemperature;
        }
    }

    public SensorState TemperatureIndicator { get; set; }

    public Sensor(string iotHubUri, string deviceId, string deviceKey)
    {
        _IotHubUri = iotHubUri;
```

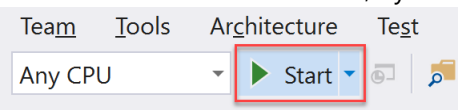
```
        DeviceId = deviceId;
        DeviceKey = deviceKey;
        State = DeviceState.Registered;
    }

    public void InstallDevice(string statusWindow)
    {
        StatusWindow = statusWindow;
        State = DeviceState.Installed;
    }

    /// <summary>
    /// Connect a device to the IoT Hub by instantiating a DeviceClient for that Device by
Id and Key.
    /// </summary>
    public void ConnectDevice()
    {
        //TODO: 17. Connect the Device to Iot Hub by creating an instance of DeviceClient
        _DeviceClient = DeviceClient.Create(_IotHubUri, new
DeviceAuthenticationWithRegistrySymmetricKey(DeviceId, DeviceKey));

        //Set the Device State to Ready
        State = DeviceState.Ready;
    }

    public void DisconnectDevice()
    {
        //Delete the local device client
        _DeviceClient = null;

        //Set the Device State to Activate
        State = DeviceState.Activated;
    }

    /// <summary>
    /// Send a message to the IoT Hub from the Smart Meter device
    /// </summary>
    public async void SendMessageAsync()
    {
        var telemetryDataPoint = new
        {
            id = DeviceId,
            time = DateTime.UtcNow.ToString("o"),
            temp = CurrentTemperature
        };

        //TODO: 18.Serialize the telemetryDataPoint to JSON
        var messageString = JsonConvert.SerializeObject(telemetryDataPoint);

        //TODO: 19.Encode the JSON string to ASCII as bytes and create new Message with the
bytes
        var message = new Message(Encoding.ASCII.GetBytes(messageString));
```

```
        //TODO: 20.Send the message to the IoT Hub
        var sendEventAsync = _DeviceClient?.SendEventAsync(message);
        if (sendEventAsync != null) await sendEventAsync;
    }
}
```

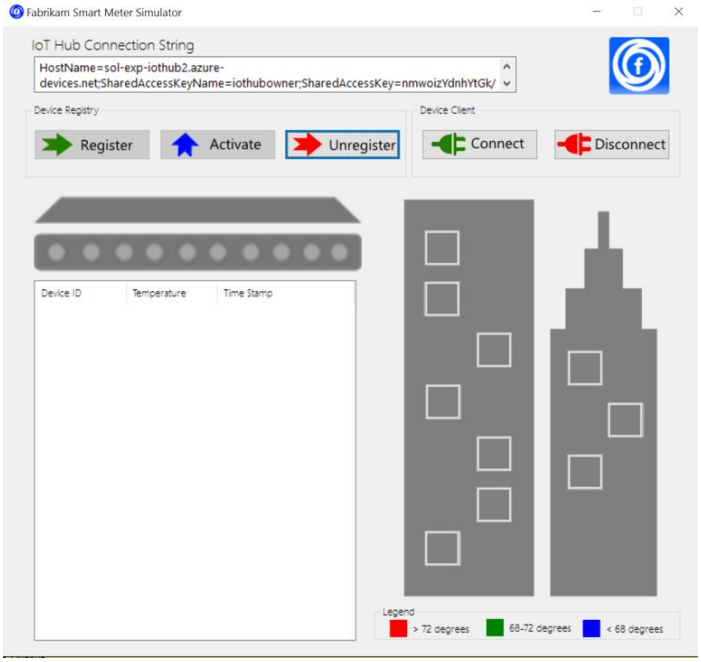
3. Save Sensor.cs

Task 3: Verify device registration and telemetry

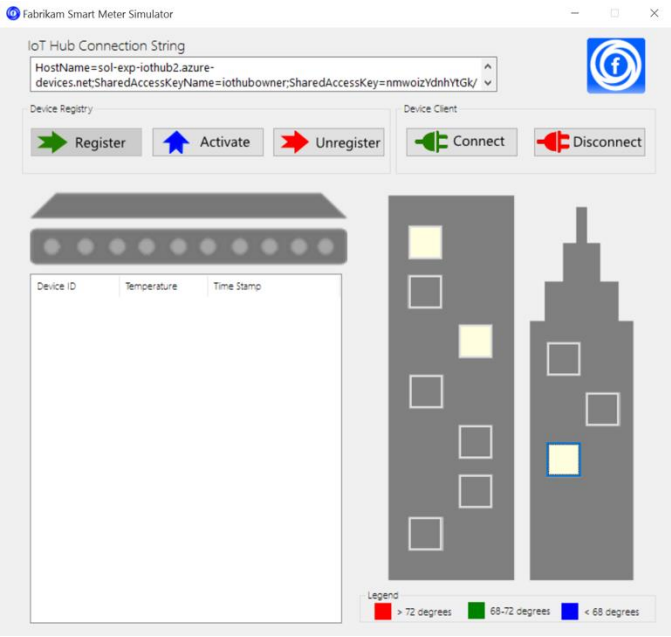
1. Run the Smart Meter Simulator, by clicking on the green Start button on the Visual Studio menu bar.



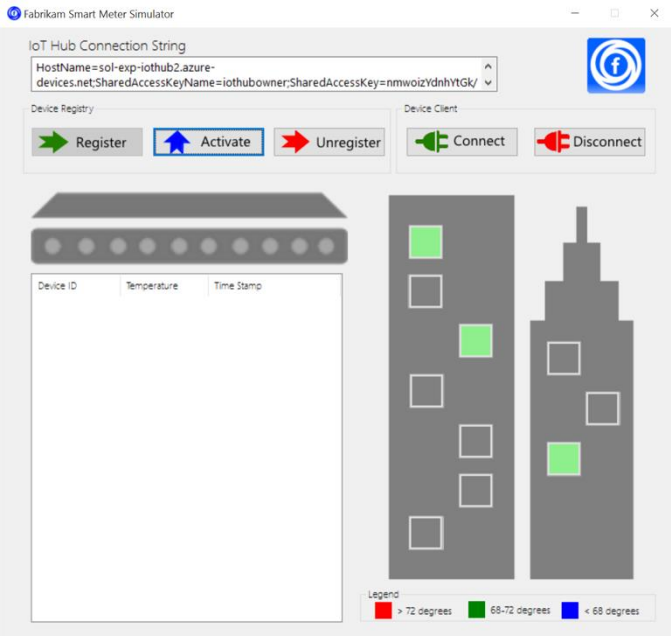
2. Click **Register**. The windows within the building should turn from black to gray.



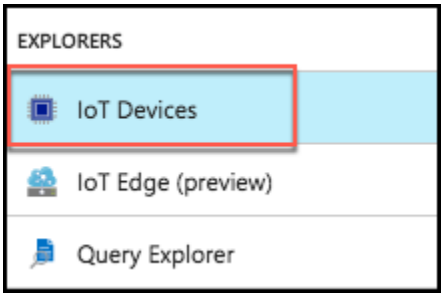
3. Click some of the windows. Each represents a device for which you want to simulate device installation. The selected windows should turn white.



4. Click **Activate** to simulate changing the device status from disabled to enabled in the Iot Hub Registry. The selected windows should turn green.



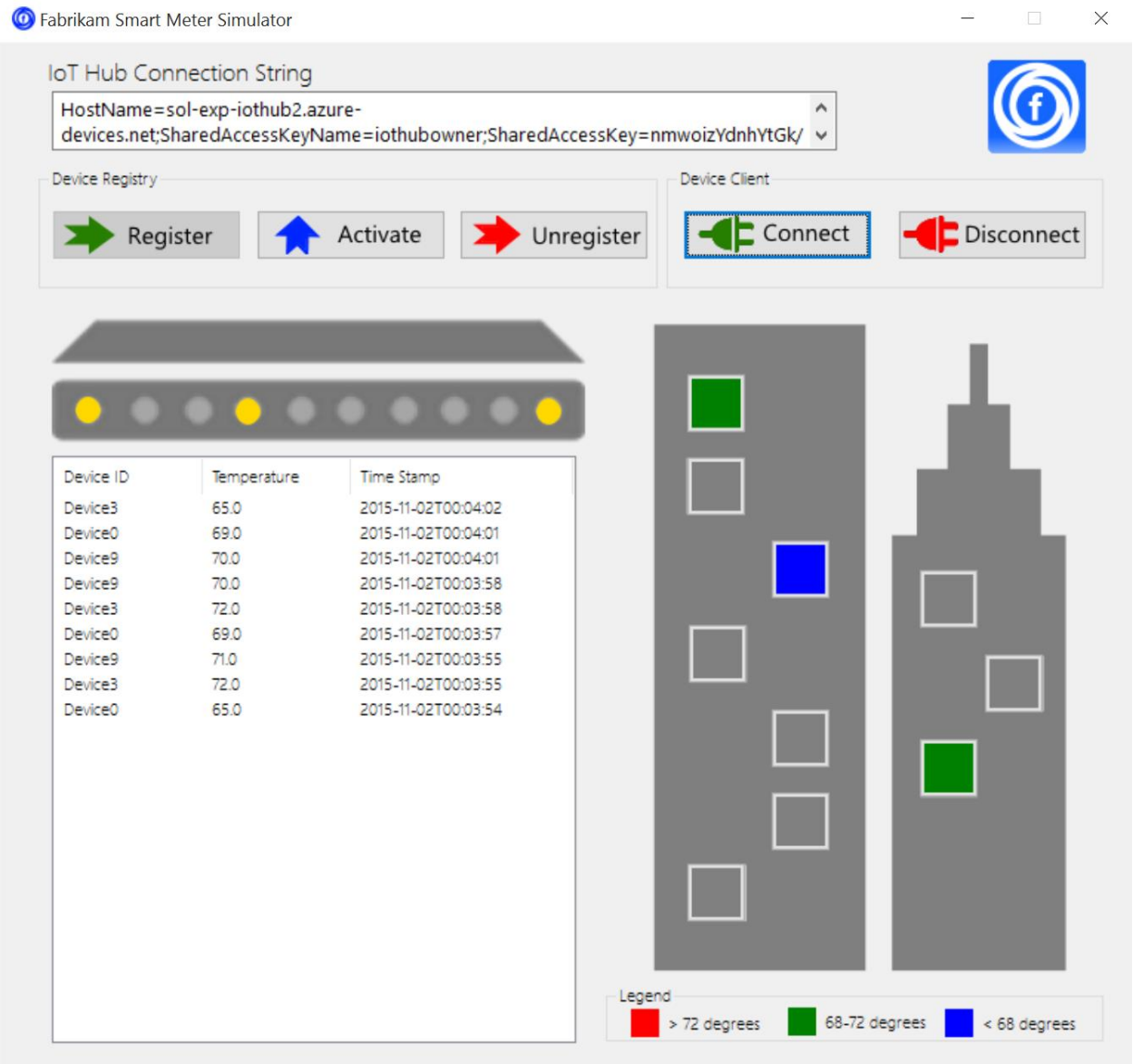
5. At this point, you have registered 10 devices (the gray windows) but activated only the ones you selected (in green). To view this list of devices, you will switch over to the Azure Portal, and open the IoT Hub you provisioned.
6. From the IoT Hub blade, click the **IoT Devices** link under Explorers on the left-hand menu.



7. You should see all 10 devices listed, with the ones that you activated having a status of **enabled**.

DEVICE ID	STATUS
Device0	enabled
Device1	enabled
Device2	disabled
Device3	enabled
Device4	disabled
Device5	disabled
Device6	disabled
Device7	disabled
Device8	enabled
Device9	enabled

8. Return to the **Smart Meter Simulator**.
9. Click **Connect**. Within a few moments, you should begin to see activity as the windows change color indicating the smart meters are transmitting telemetry. The grid on the left will list each telemetry message transmitted and the simulated temperature value.

10. 

11. Allow the smart meter to continue to run. (Whenever you want to stop the transmission of telemetry, click the **Disconnect** button.)

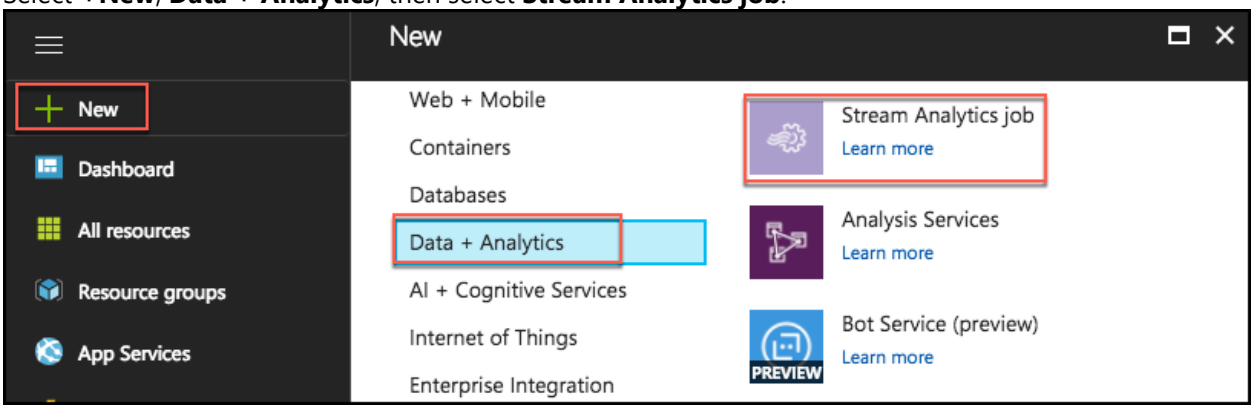
Exercise 4: Hot path data processing with Stream Analytics

Duration: 45 minutes

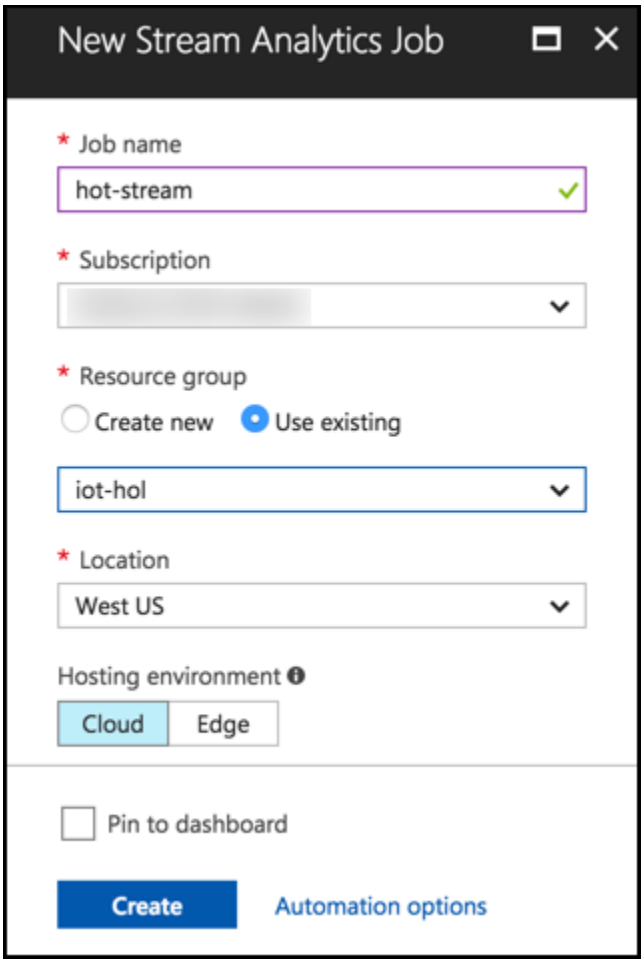
Fabrikam would like to visualize the "hot" data showing the average temperature reported by each device over a 5-minute window in Power BI.

Task 1: Create a Stream Analytics job for hot path processing to Power BI

1. In your browser, navigate to the **Azure Portal** (https://portal.azure.com).
2. Select **+New**, **Data + Analytics**, then select **Stream Analytics job**.

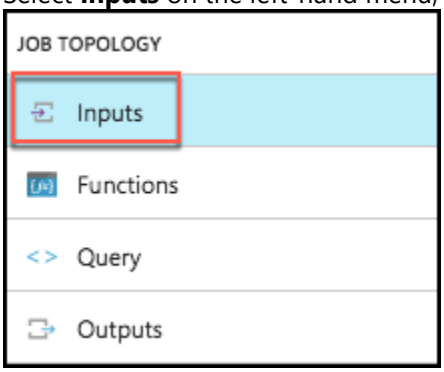


3. On the New Stream Analytics Job blade, enter the following:
   a. Job Name: Enter **hot-stream**.
   b. Subscription: Choose the same subscription you have been using thus far.
   c. Resource Group: Choose the **iot-hol** Resource Group.
   d. Location: Choose the same Location as you have for your other resources.
   e. Hosting environment: Select **Cloud**.

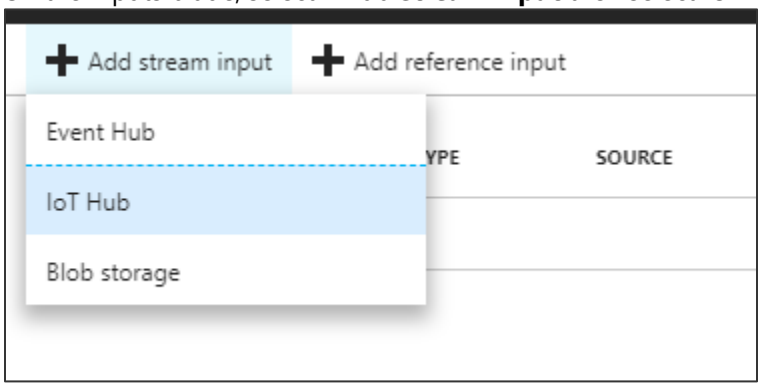


f. Select **Create** to provision the new Stream Analytics job.

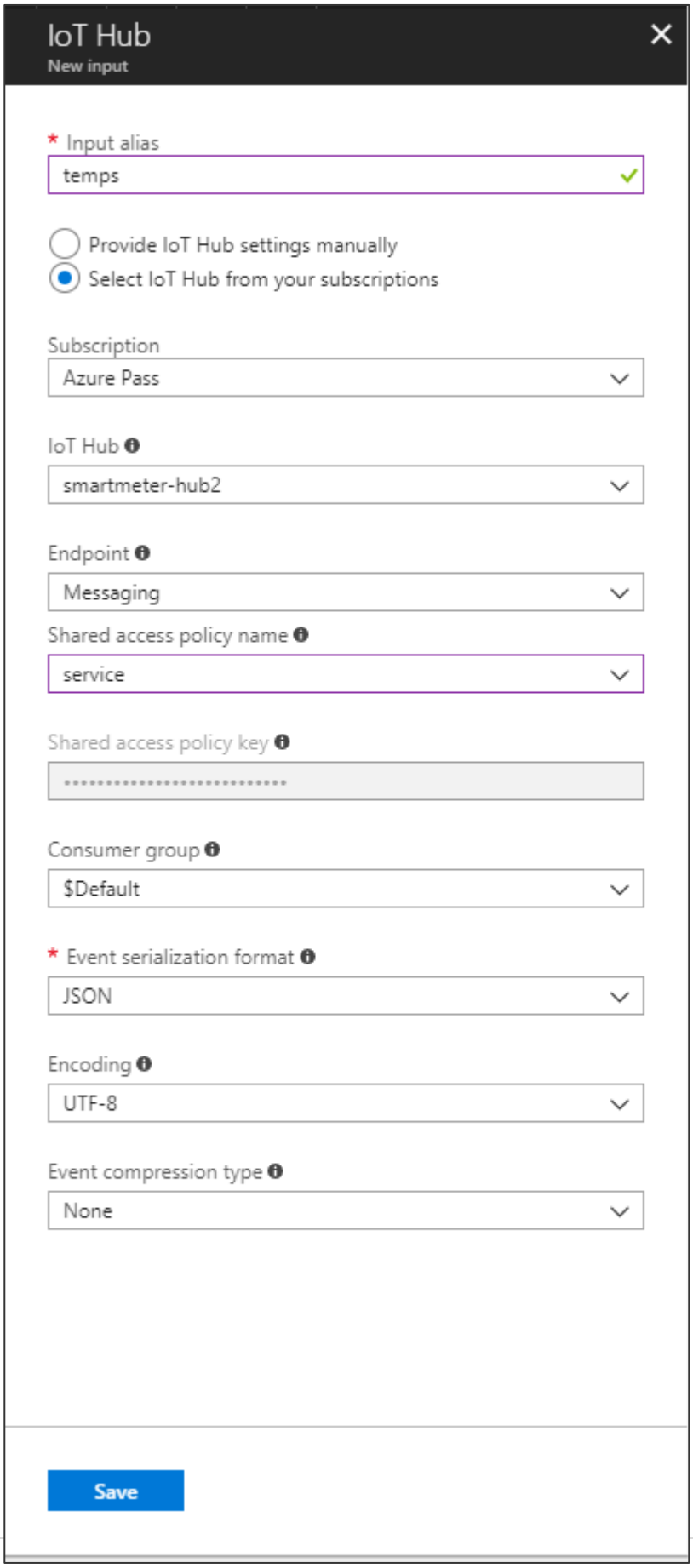
4. Once provisioned, navigate to your new Stream Analytics job in the portal.
5. Select **Inputs** on the left-hand menu, under Job Topology.



6. On the Inputs blade, select **+Add stream input** then select **IoT Hub** to add an input connected to your IoT Hub.

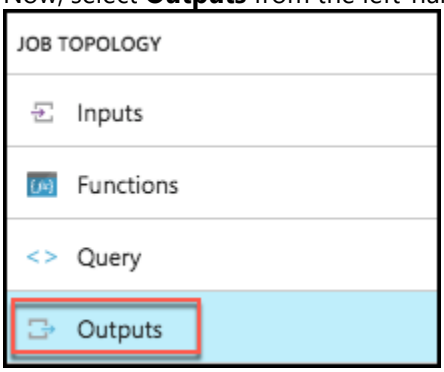


7. On the New Input blade, enter the following:
    a. Input Alias: Set the value to **temps**.
    b. Leave the Select **IoT hub from your subscriptions** option selected
    c. IoT Hub: Select your existing IoT Hub, eg: **smartmeter-hub**.
    d. Endpoint: Choose **Messaging**.
    e. Shared Access Policy Name: Select **Service**.
    f. Consumer Group: Leave as **$Default**.
    g. Event serialization format: Choose **JSON**.
    h. Encoding: Choose **UTF-8**.
    i. Event compression type: Leave set to **None**.

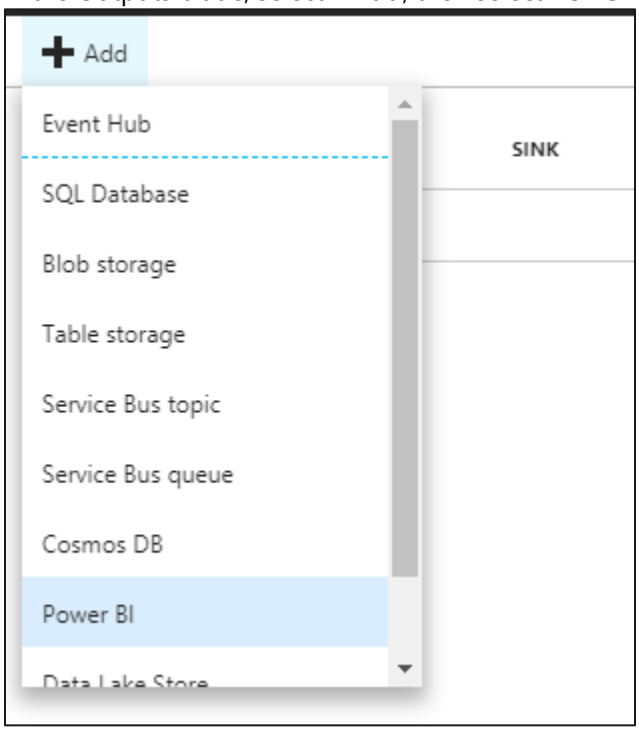


j. Select **Save**.

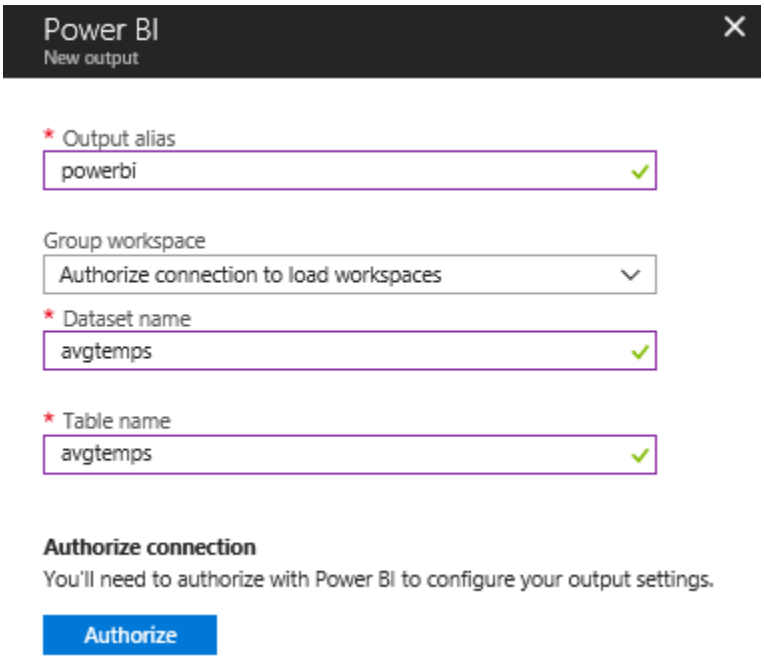
8. Now, select **Outputs** from the left-hand menu, under Job Topology.



9. In the Outputs blade, select **+Add**, then select **Power BI**, to add the output destination for the query.

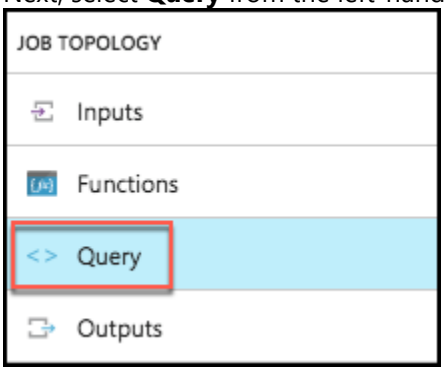


10. On the New output blade, enter the following:
    a. Output alias: Set to **powerbi**.
    b. Group workspace: Select **Authorize connection to load workspaces**
    c. Dataset Name: Set to **avgtemps**
    d. Table Name: Set to **avgtemps**
    e. Select **Authorize** to authorize the connection to your Power BI account. When prompted in the popup window, enter the account credentials you used to create your Power BI account in Before the Hands-on Lab, Task 1.



f. Select **Save**.

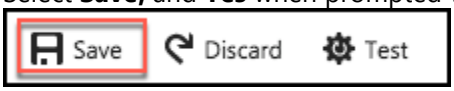
11. Next, select **Query** from the left-hand menu, under Job Topology.



12. In the query text box, paste the following query.

```
SELECT AVG(temp) AS Average, id
INTO powerbi
FROM temps
GROUP BY TumblingWindow(minute, 5), id
```

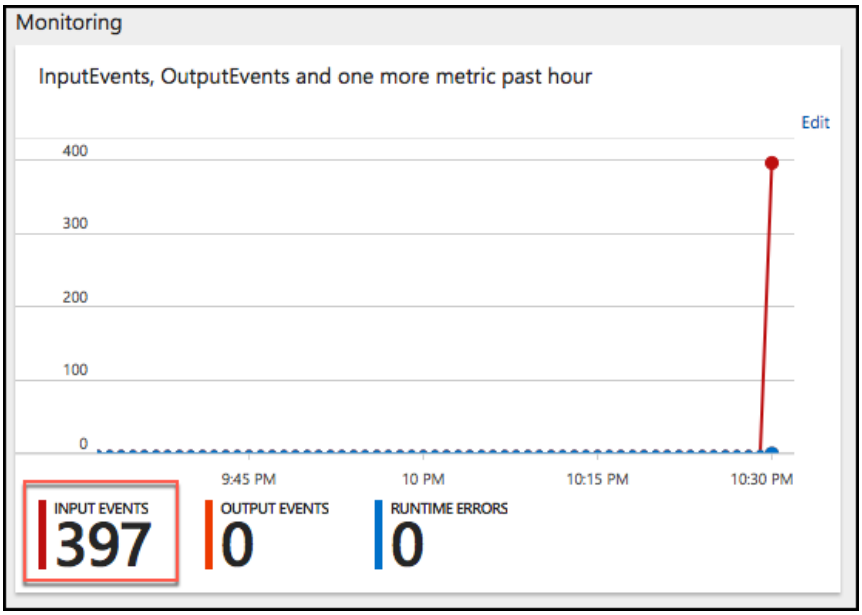
13. Select **Save,** and **Yes** when prompted with the confirmation.



14. Return to the Overview blade on your Stream Analytics job, and select **Start**.

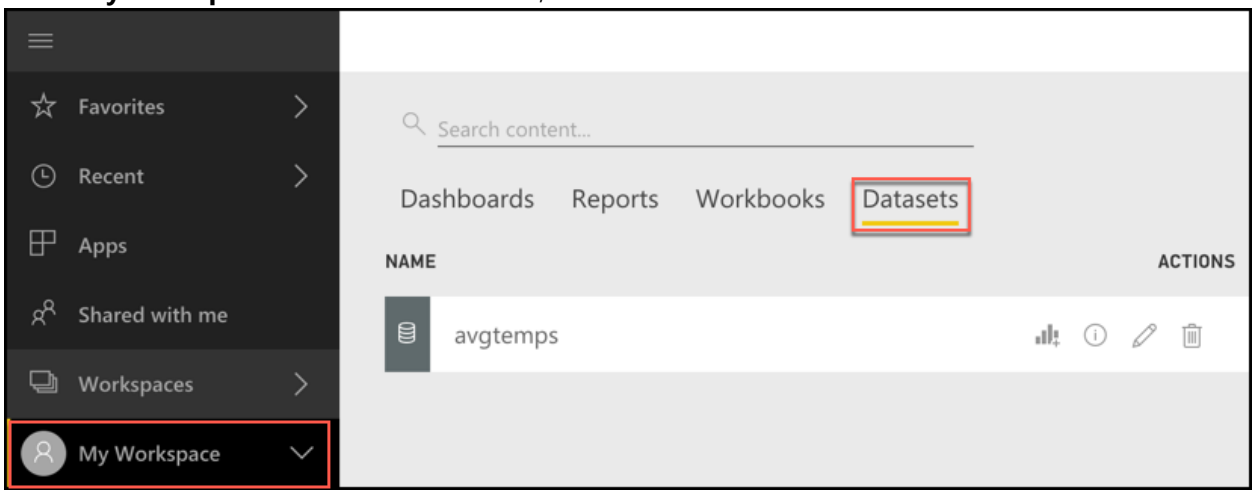
15. In the Start job blade, select **Now** (the job will start processing messages from the current point in time onward).

16. Select **Start**.
17. Allow your Stream Analytics Job a few minutes to start.
18. Once the Stream Analytics Job has successfully started, verify that you are showing a non-zero amount of **Input Events** on the **Monitoring** chart on the overview blade. You may need to reconnect your devices on the Smart Meter Simulator and let it run for a while to see the events.

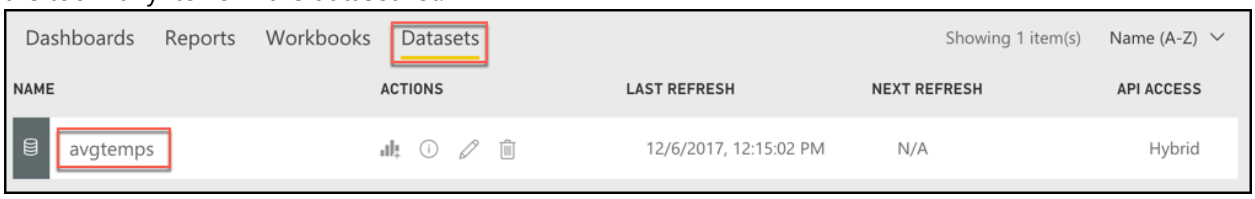


Task 2: Visualize hot data with Power BI

1. Sign in to your Power BI subscription (https://app.powerbi.com) to see if data is being collected.
2. Select **My Workspace** on the left-hand menu, then select the **Datasets** tab.

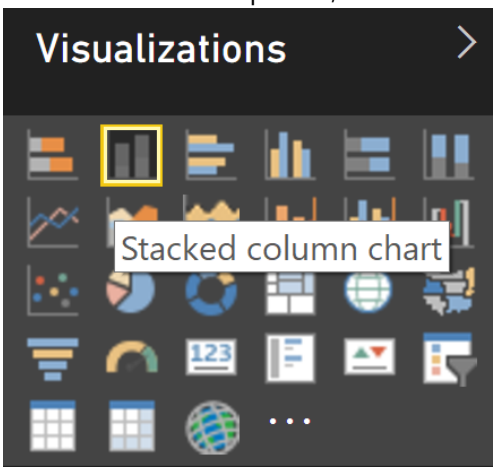


3. Under the Datasets list the **avgtemps** dataset should be available. You can search for the avgtemps dataset if there are too many items in the dataset list.

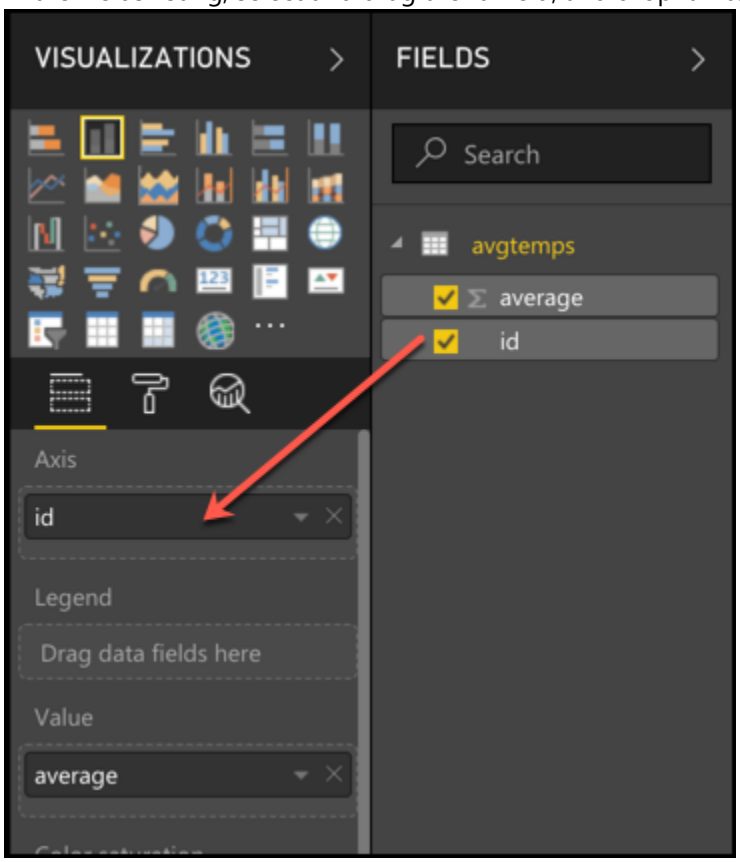


4. Select the **Create Report** button under the **Actions** column for the **avgtemps** dataset.

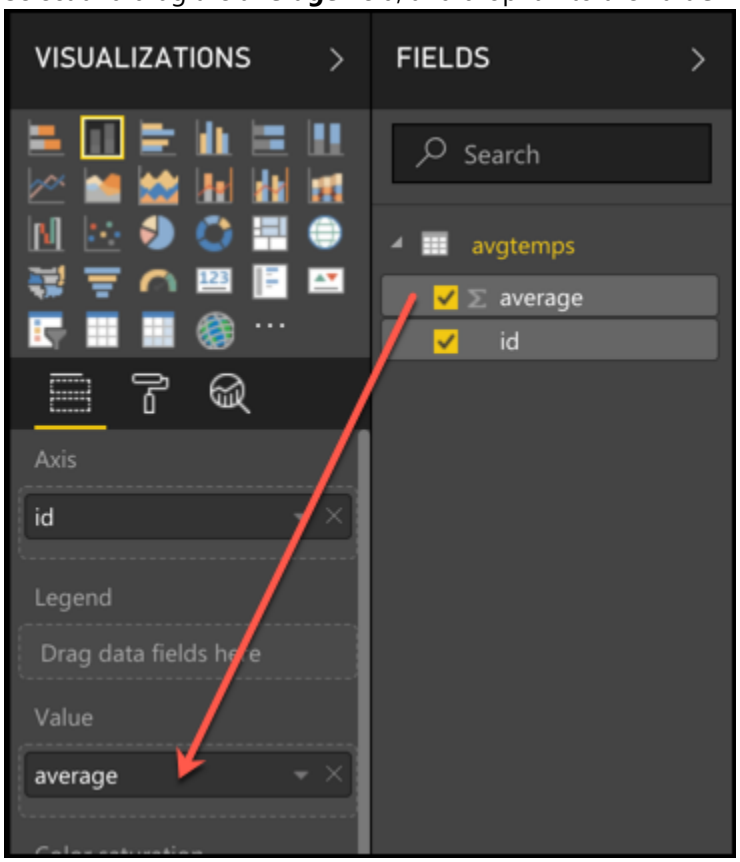
5. On the Visualizations palette, select **Stacked Column Chart** to create a chart visualization.



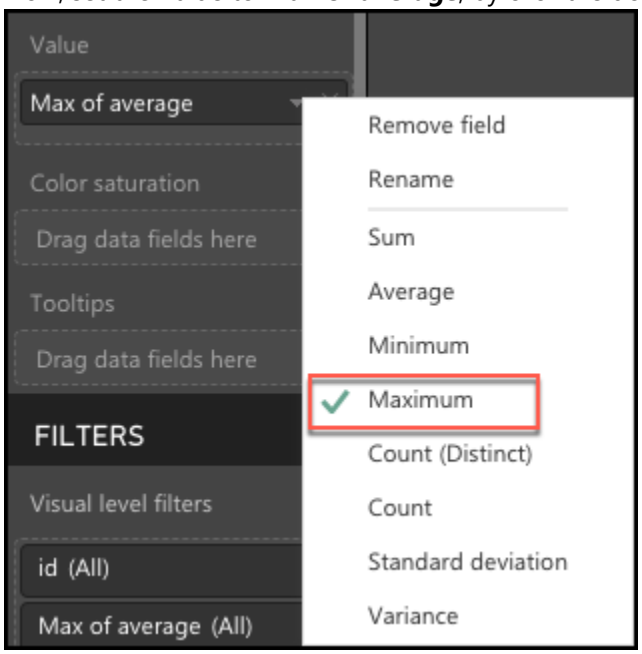
6. In the Fields listing, select and drag the **id** field, and drop it into the **Axis** field.



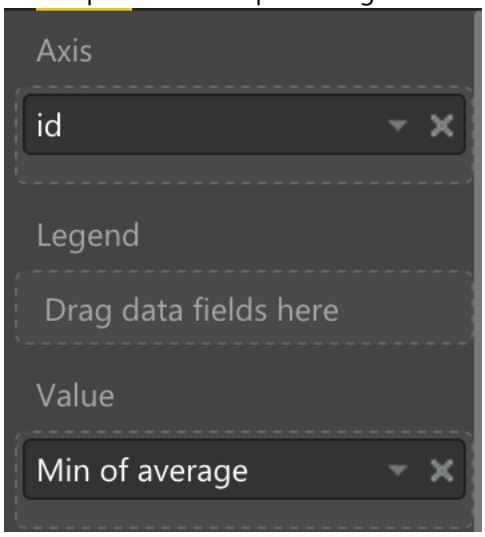
7. Select and drag the **average** field, and drop it into the **value** field.



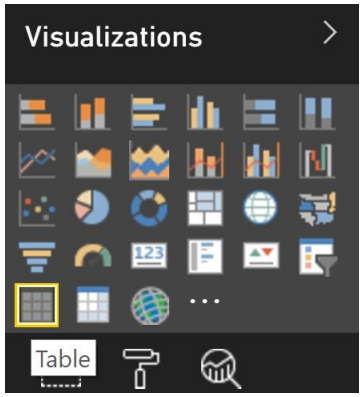
8. Now, set the Value to **Max of average**, by click the down arrow next to **average**, and select **Maximum**.



9. Repeat steps 5–8, this time adding a Stacked Column Chart for **Min of average**. (You may need to click on any area of white space on the report designer surface to deselect the Max of average chart visualization.)

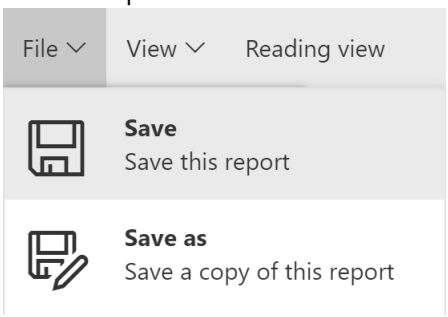


10. Next, add a **table visualization**.

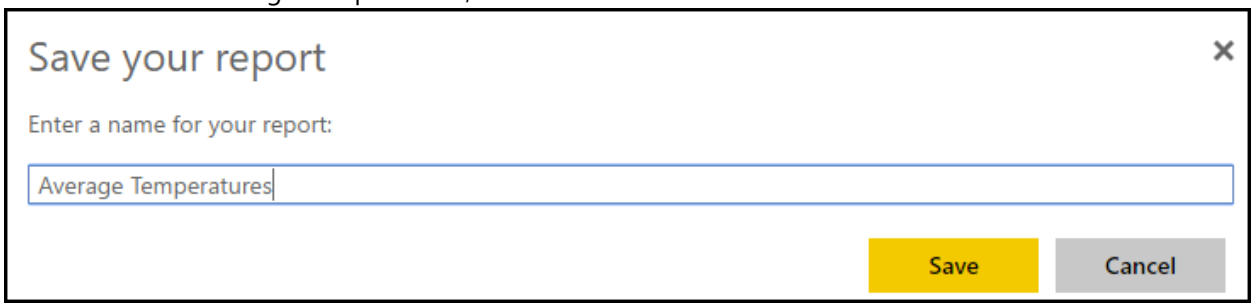


11. Set the values to **id** and **Average of average**, by dragging and dropping both fields in the Values field, then selecting the dropdown next to average, and selecting Average.

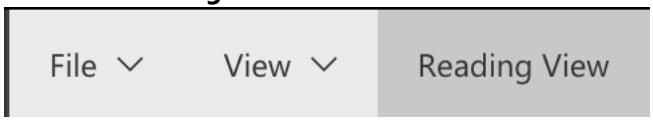
12. Save the report.



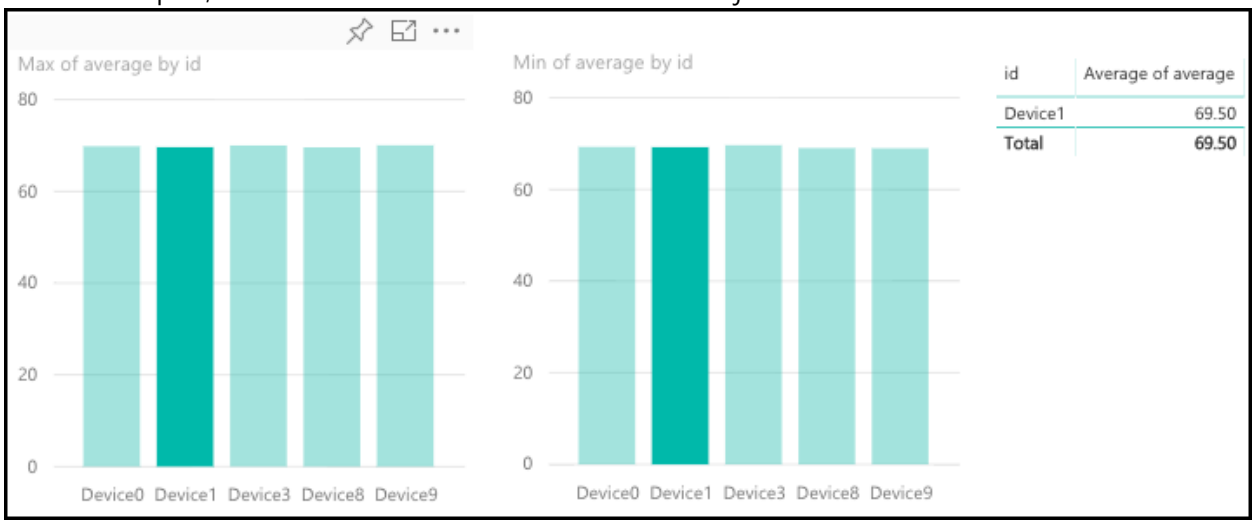
13. Enter the name "Average Temperatures," and click Save.



14. Switch to **Reading View**.



15. Within the report, click one of the columns to see the data for just that device.



Exercise 5: Cold path data processing with HDInsight Spark

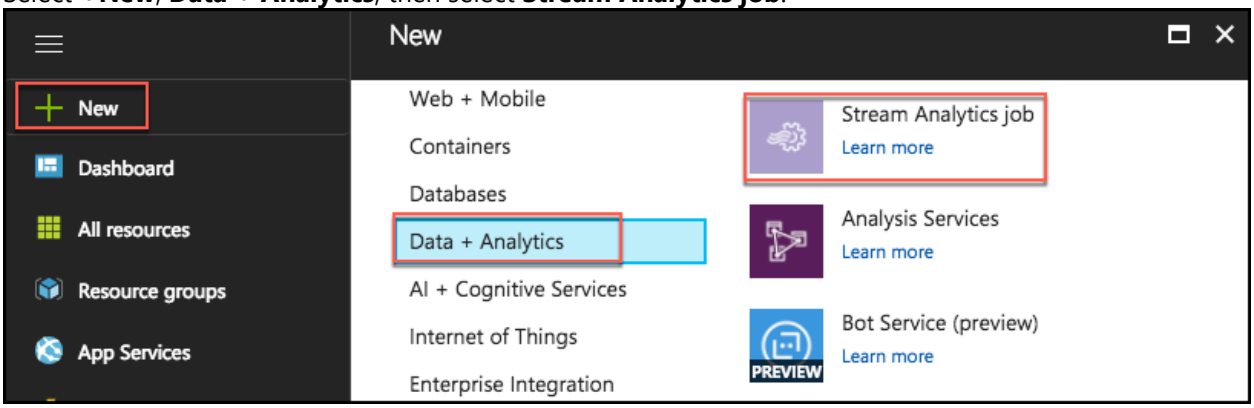
Duration: 60 minutes

Fabrikam would like to be able to capture all the "cold" data into scalable storage so that they can summarize it periodically using a Spark SQL query.

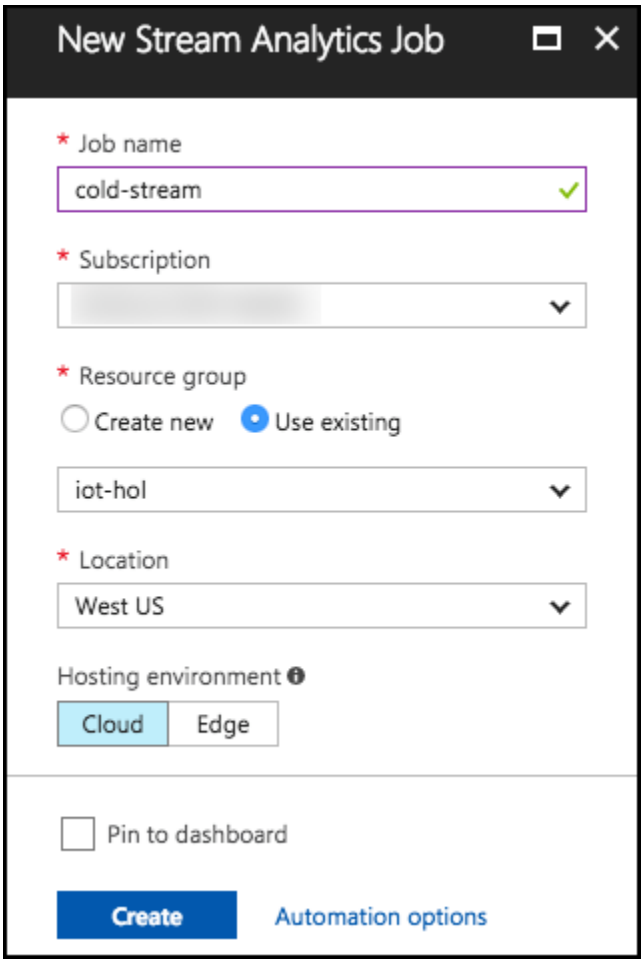
Task 1: Create the Stream Analytics job for cold path processing

To capture all metrics for the cold path, set up another Stream Analytics job that will write all events to Blob storage for analyses by Spark running on HDInsight.

1. In your browser, navigate to the **Azure Portal** (https://portal.azure.com).
2. Select **+New**, **Data + Analytics**, then select **Stream Analytics job**.

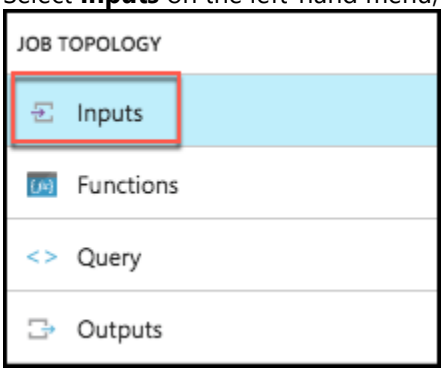


3. On the New Stream Analytics Job blade, enter the following:
   a. Job Name: Enter **cold-stream**.
   b. Subscription: Choose the same subscription you have been using thus far.
   c. Resource Group: Choose the **iot-hol** Resource Group.
   d. Location: Choose the same Location as you have for your other resources.
   e. Hosting environment: Select **Cloud**.

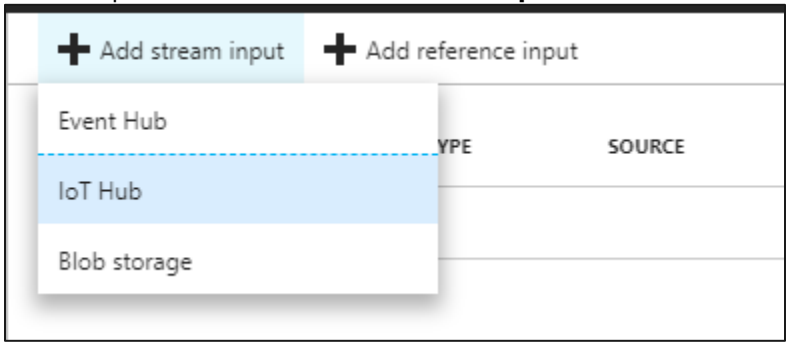


f. Select **Create**.

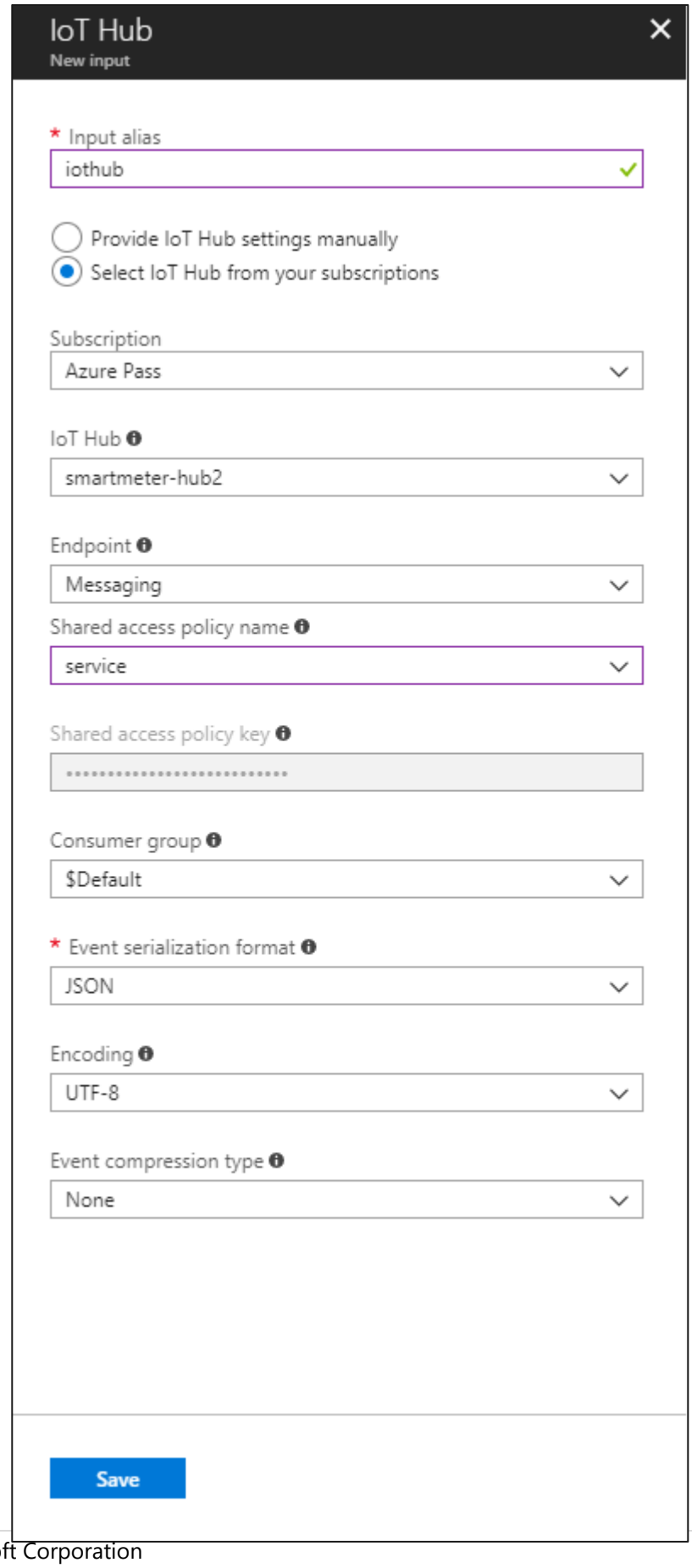
4. Once provisioned, navigate to your new Stream Analytics job in the portal.
5. Select **Inputs** on the left-hand menu, under Job Topology.



6. On the Inputs blade, select **+Add stream input**, then select **IoT Hub** to add an input connected to your IoT Hub.

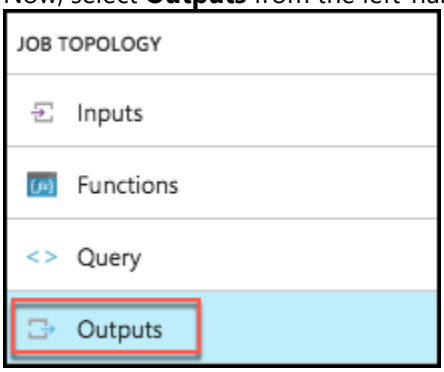


7. On the New Input blade, enter the following:
    a. Input Alias: Set the value to **iothub**.
    b. Leave the **Select IoT hub from your subscriptions** option selected.
    c. IoT Hub: Select your existing IoT Hub, eg: **smartmeter-hub**.
    d. Endpoint: Choose **Messaging**.
    e. Shared Access Policy Name: Set to **Service**.
    f. Consumer Group: Leave as **$Default**.
    g. Event serialization format: Choose **JSON**.
    h. Encoding: Choose **UTF-8**.
    i. Event compression type: Leave set to **None**.

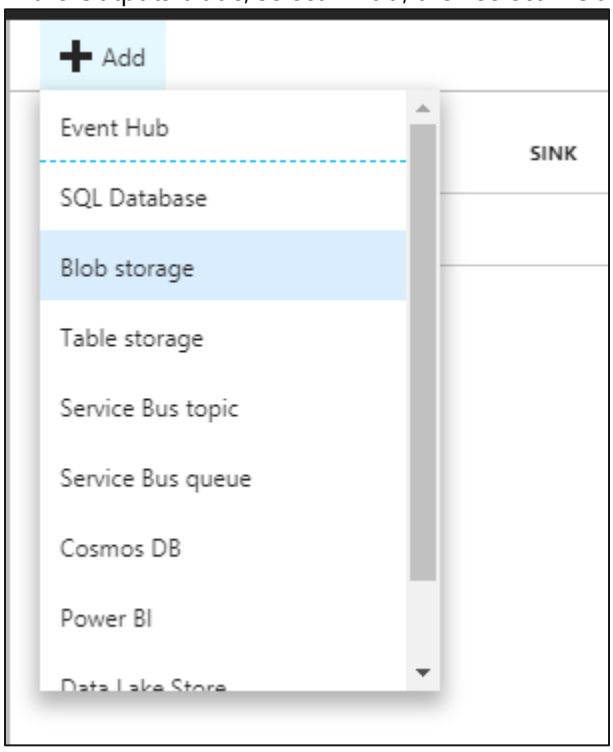

IoT Hub
New input
* Input alias
iothub
Provide IoT Hub settings manually
Select IoT Hub from your subscriptions
Subscription
Azure Pass
IoT Hub
smartmeter-hub2
Endpoint
Messaging
Shared access policy name
service
Shared access policy key
Consumer group
$Default
* Event serialization format
JSON
Encoding
UTF-8
Event compression type
None
Save

j. Select **Save**.

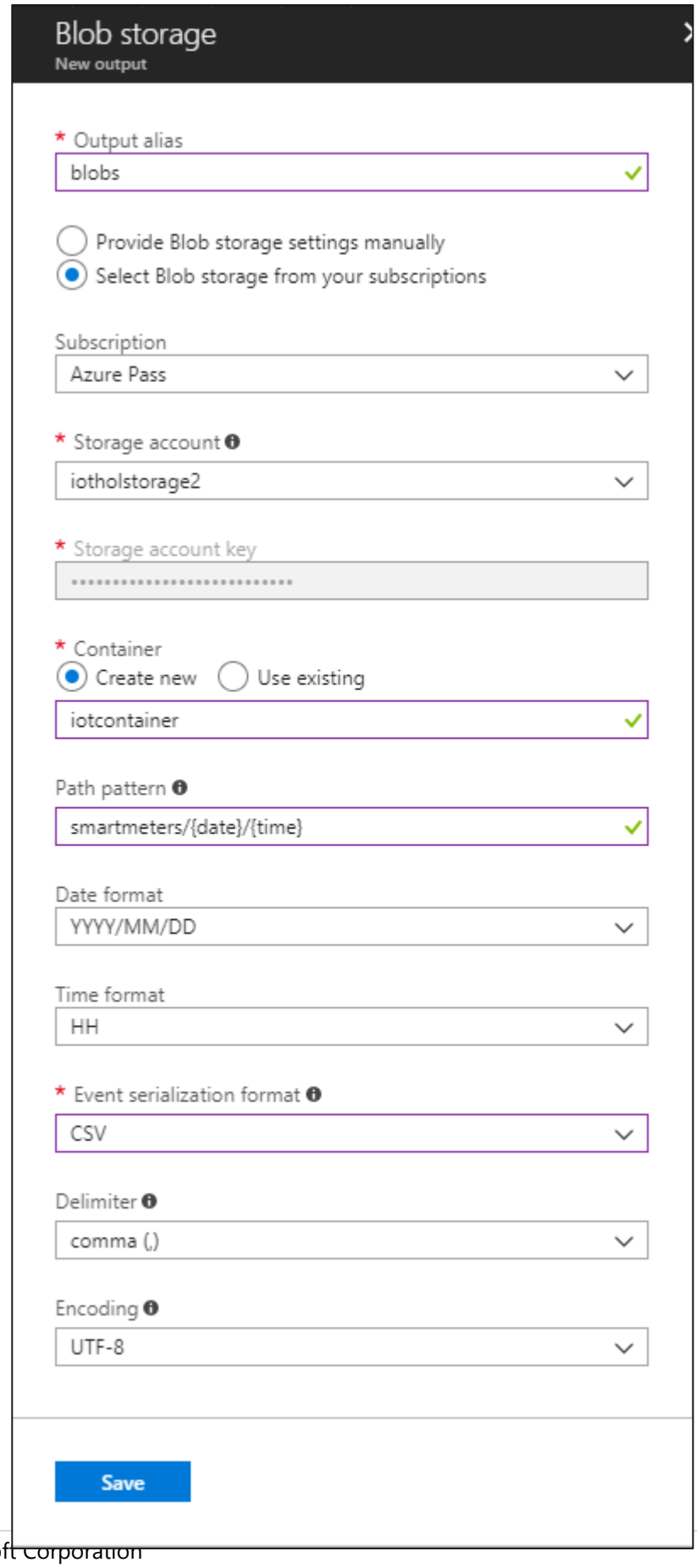
8. Now, select **Outputs** from the left-hand menu, under Job Topology.



9. In the Outputs blade, select **+Add**, then select **Blob storage** to add the output destination for the query.

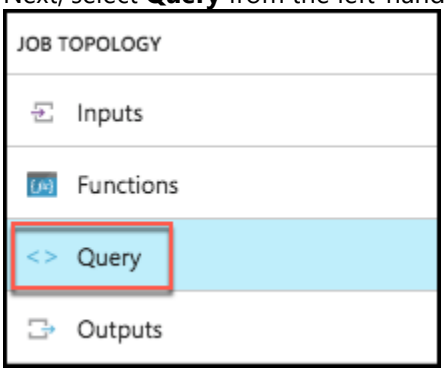


10. On the New output blade, enter the following:
    a. Output alias: Set to **blobs**.
    b. Leave the **Select blob storage from your subscriptions** option selected.
    c. Storage account: Choose the storage account name you used for HDInsight in Before the hands-on lab, Task 2, Step 5c.
    d. Container: Set to **iotcontainer**.
    e. Path pattern: Enter **smartmeters/{date}/{time}**
    f. Date format: Select **YYYY/MM/DD**.
    g. Time format: Select **HH**.
    h. Event serialization formation: Select **CSV**.
    i. Delimiter: Select **comma (,)**.
    j. Encoding: Select **UTF-8**.



k. Select **Save**.

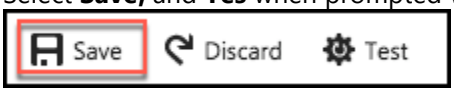
19. Next, select **Query** from the left-hand menu, under Job Topology.



20. In the query text box, paste the following query.

```
SELECT
      *
INTO
      blobs
FROM
      iothub
```

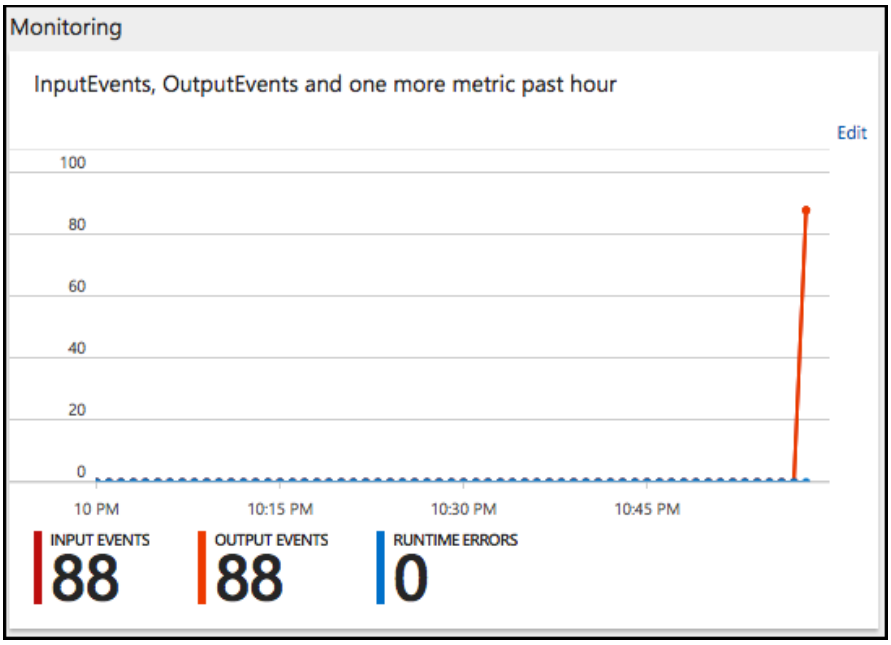
21. Select **Save,** and **Yes** when prompted with the confirmation.



22. Return to the Overview blade on your Stream Analytics job, and select **Start**.

23. In the Start job blade, select **Now** (the job will start processing messages from the current point in time onward).

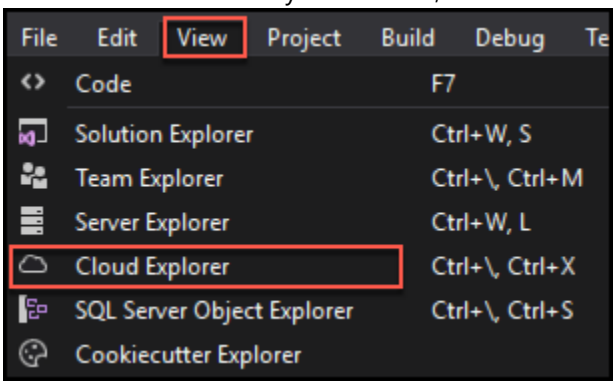
24. Select Start.
25. Allow your Stream Analytics Job a few minutes to start.
26. Once the Stream Analytics Job has successfully started, verify that you are showing a non-zero amount of **Input Events** on the **Monitoring** chart on the overview blade. You may need to reconnect your devices on the Smart Meter Simulator and let it run for a while to see the events.



Task 2: Verify CSV files in blob storage

In this task, we are going to verify that the CSV file is being written to blob storage. (Note, this can be done via Visual Studio, or using the Azure portal. For this lab, we will perform the task using Visual Studio.)

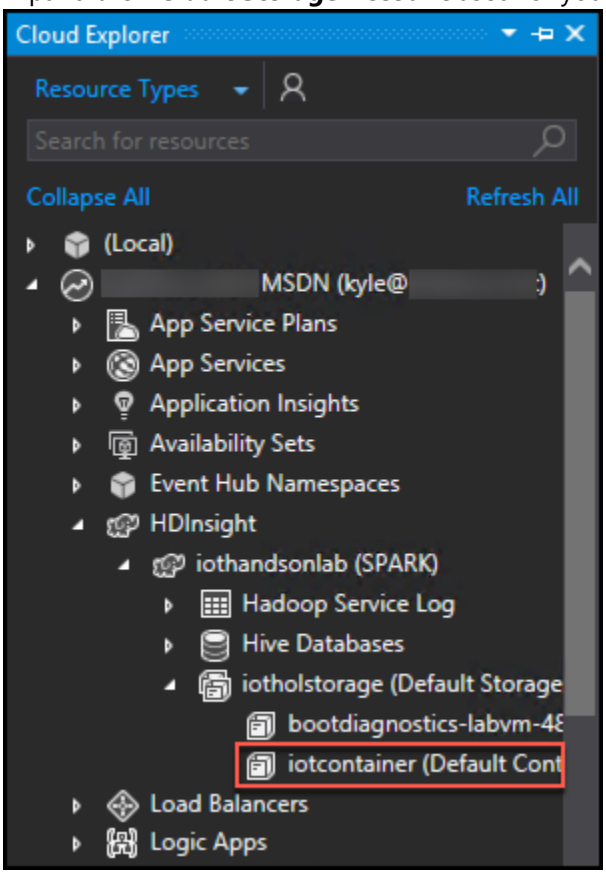
1. Within Visual Studio on your Lab VM, select the **View** menu, then select **Cloud Explorer**.



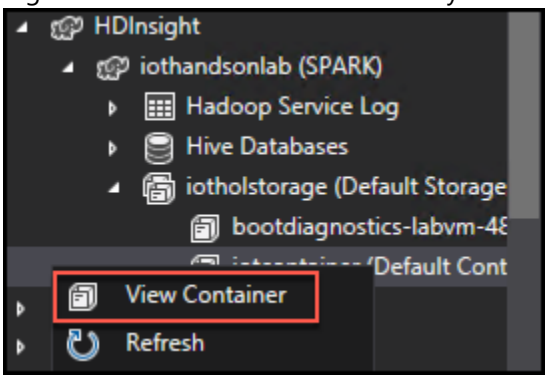
2. In **Cloud Explorer**, select Account Management, and connect to Microsoft Azure Subscription.

3. If prompted, sign into your Azure account.
4. Allow Cloud Explorer about 30 seconds to load your subscription resources.
5. Expand your Azure account, and then expand **HDInsight**. It may take a few moments to load your storage accounts.
6. Expand the Spark cluster you created for this lab.

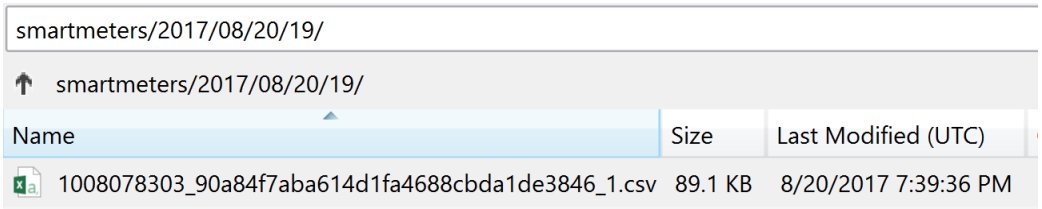
7. Expand the **Default Storage Account** used for your HDInsight cluster.



8. Right-click the container named after your HDInsight cluster. Select **View Container**.



9. Verify files are being written to Blob storage, and take note of the path to one of the files (the files should be located underneath the smartmeters folder).

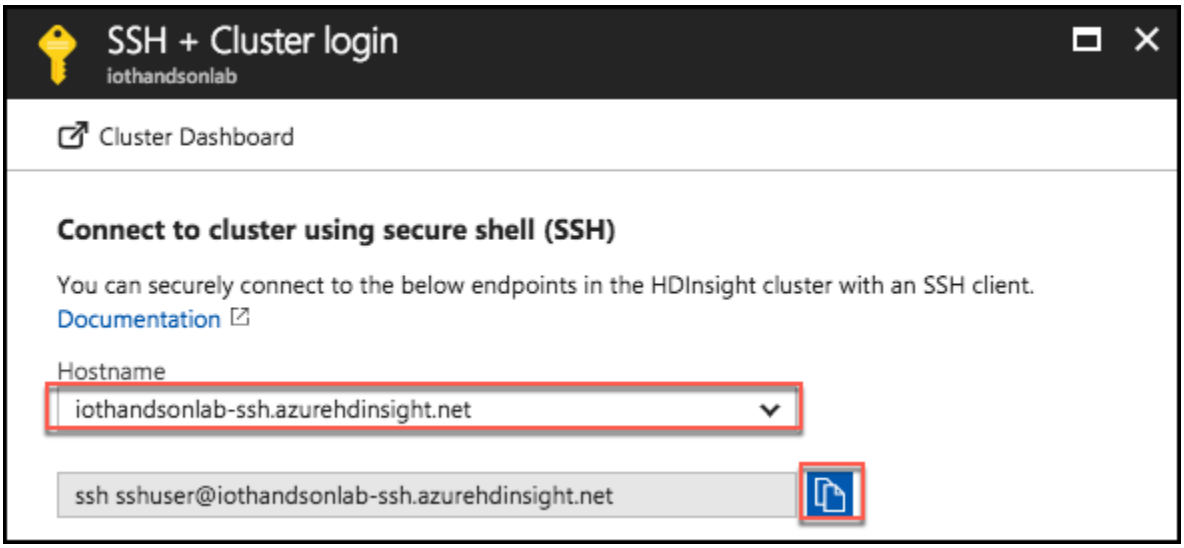


Task 3: Update pandas version on the Spark cluster

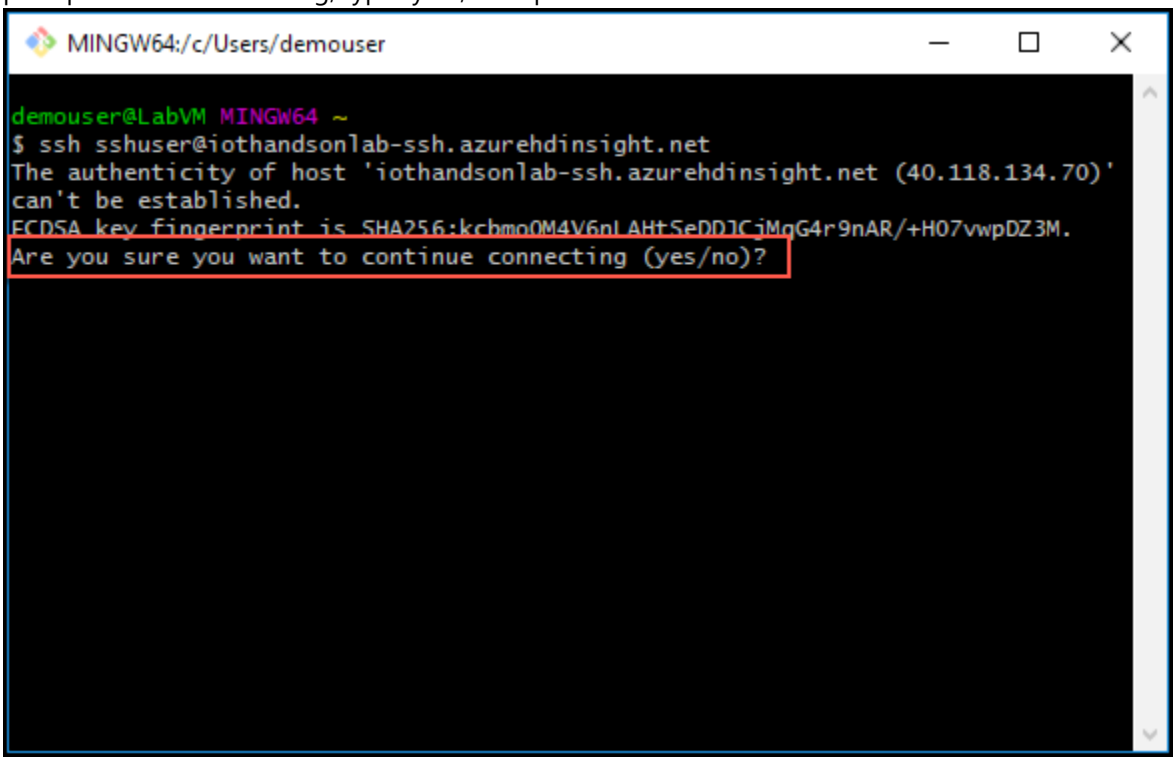
In this task you will connect SSH into your HDInsight cluster, and update the version of pandas that Jupyter Notebook uses. This task is to address errors that are displayed because the autovizwidget in Jupyter needs a later version of pandas that has the API module.

1. In the Azure portal, navigate to the blade for your Spark Cluster, under your HDInsight Cluster.
2. On the cluster's overview blade, select Secure Shell (SSH) from the toolbar.

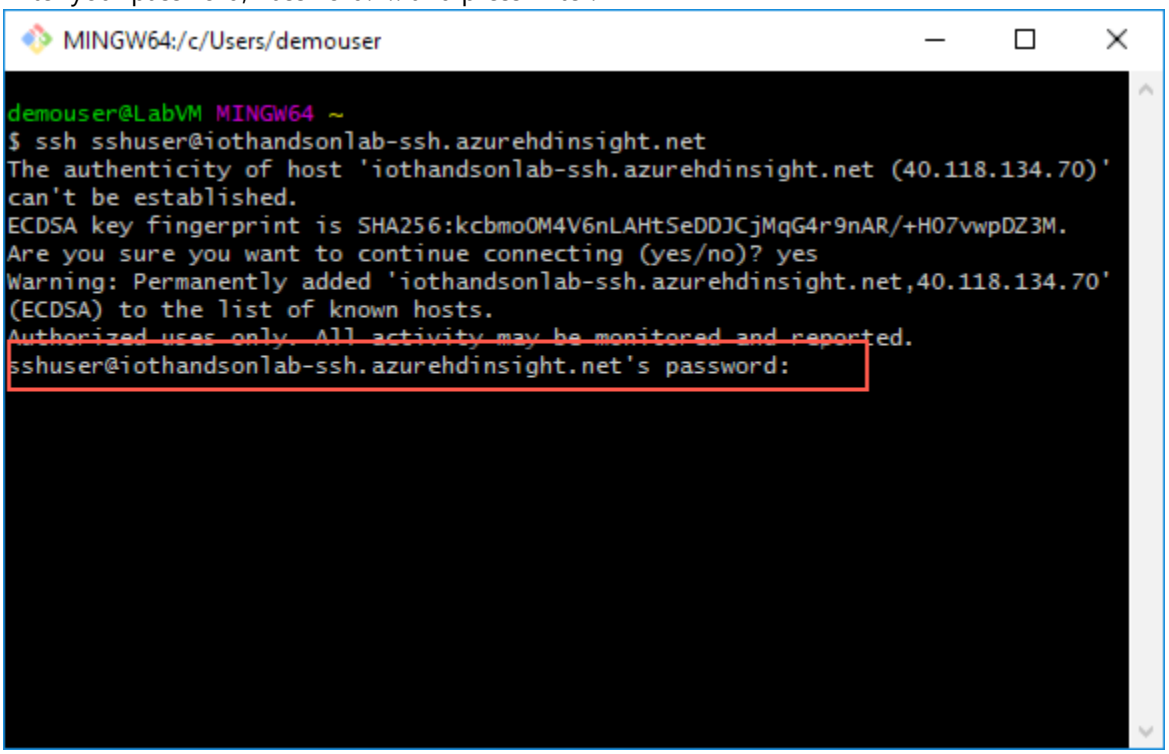
3. On the SSH + Cluster login blade, select your cluster from the Hostname drop down, then select the copy button next to the text box.



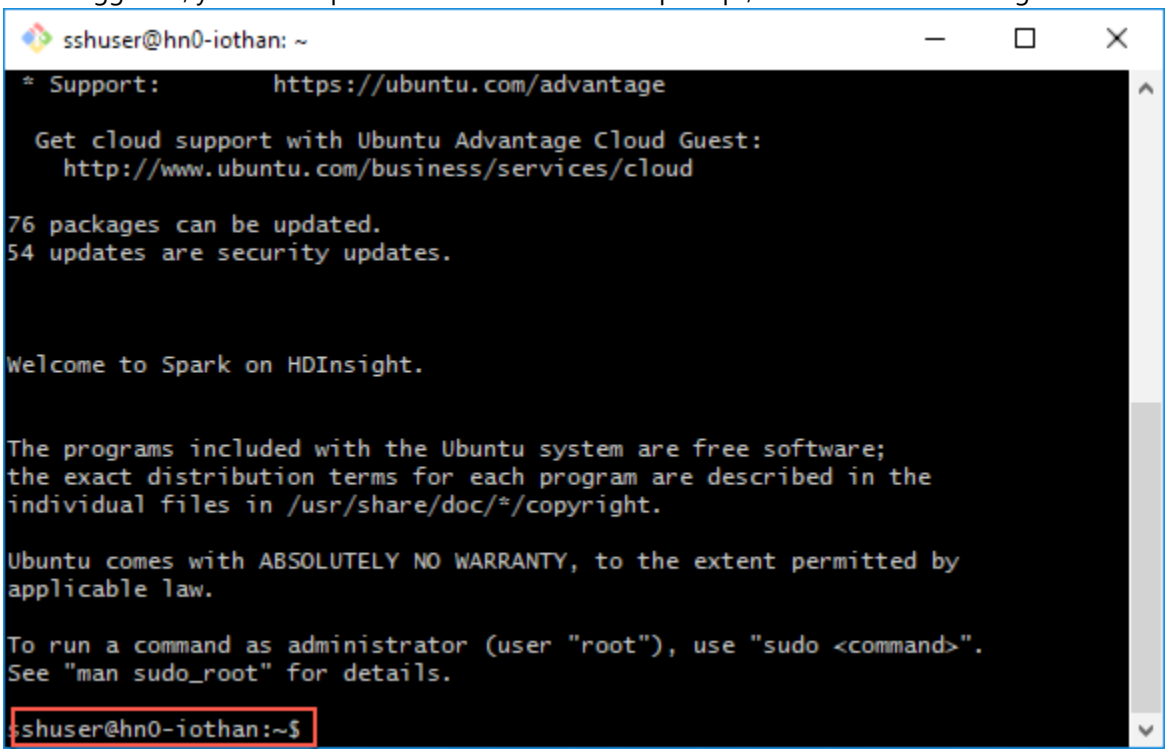
4. On your Lab VM, select your open Git Bash client (or open a new one if you closed it).
5. At the command prompt, paste the SSH connection string you copied in step 3 above, then press enter. When prompted about continuing, type "yes", then press enter.



6. Enter your password, Password.1!! and press Enter.



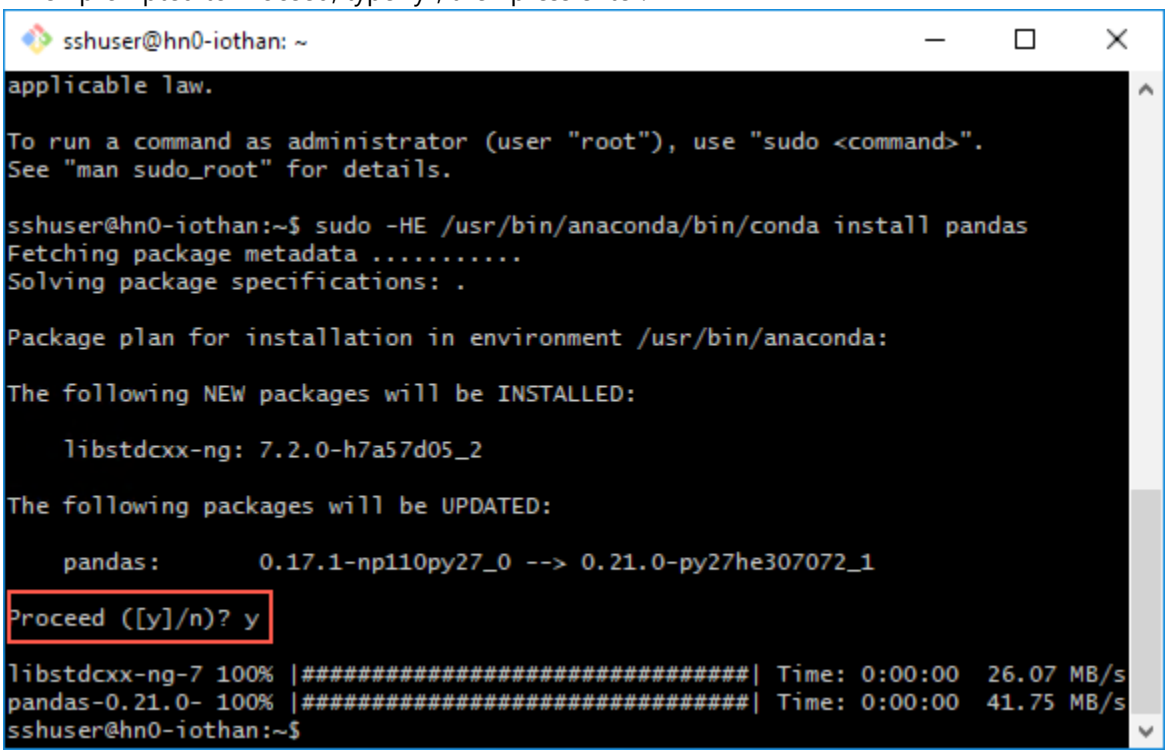
7. Once logged in, you will be presented with a command prompt, similar to the following:



8. At the command prompt, enter the following command, then press enter to install the latest version of pandas, which has the API module required by the autovizwidget.

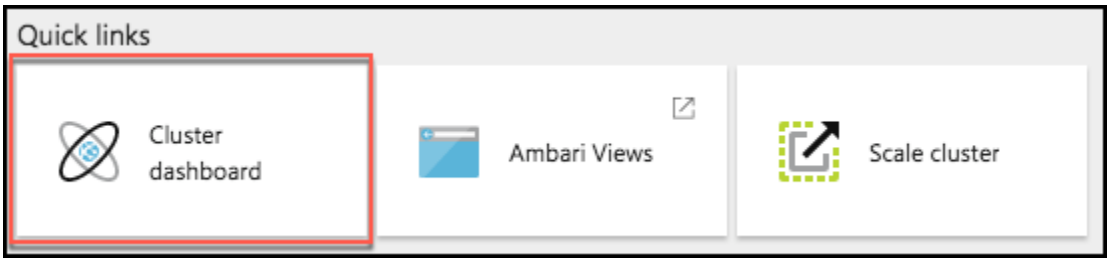
```
sudo -HE /usr/bin/anaconda/bin/conda install pandas
```

9. When prompted to Proceed, type “y”, then press enter.

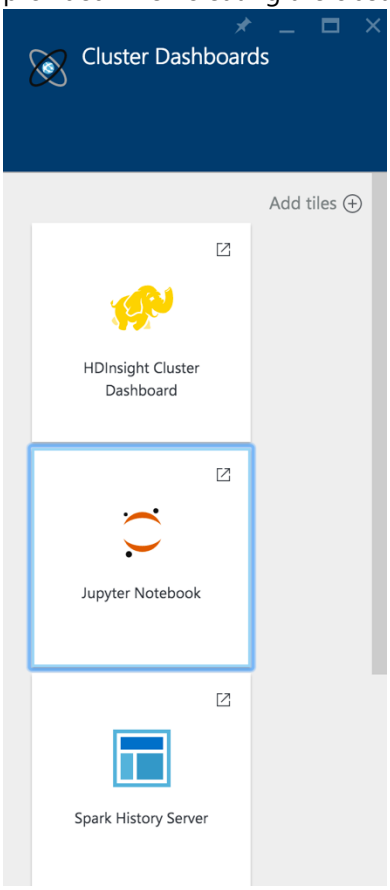


Task 4: Process with Spark SQL

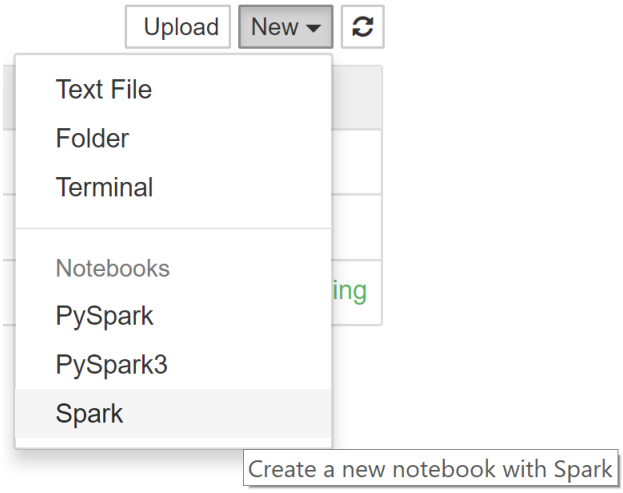
1. Navigate to the blade for your Spark Cluster in the Azure Portal, under you HDInsight Cluster.
2. Under Quick Links, click **Cluster Dashboards**.



3. On the Cluster Dashboards blade, select **Jupyter Notebook**. If prompted, log in with admin credentials you provided when creating the cluster (username: **admin**, password: **Password.1!!**).



4. From the navigation bar in the Jupyter site, select **New** and then **Spark.**

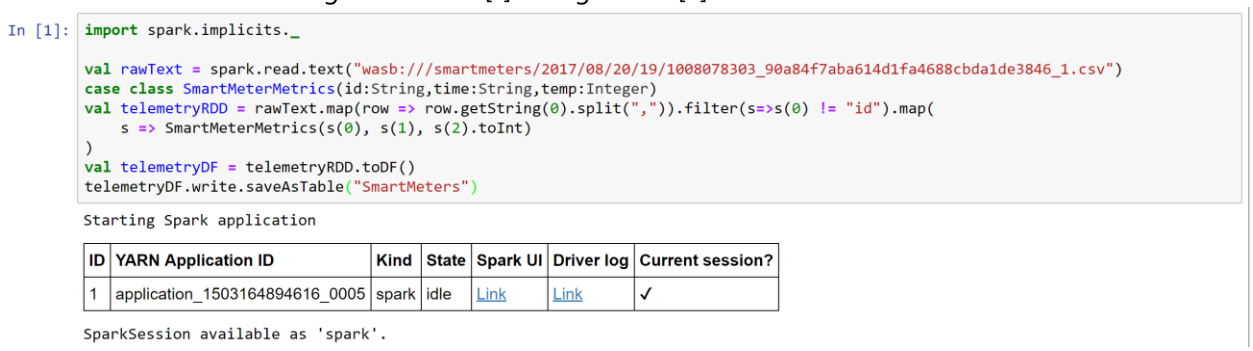


5. In the first text area (referred to as a paragraph in notebook jargon), enter the following **Scala code** that will load, parse, and save your batch scored telemetry data as a table that you can later query using Spark SQL. Before executing, make sure to replace the highlighted text with the correct path to your telemetry file, noted previously in the Cloud Explorer in Visual Studio.

```
import spark.implicits._

val rawText =
spark.read.text("wasb:///smartmeters/2017/08/20/19/1008078303_90a84f7aba614d1fa4688cbda1de3846_1.csv")
case class SmartMeterMetrics(id:String,time:String,temp:Integer)
val telemetryRDD = rawText.map(row => row.getString(0).split(",")).filter(s=>s(0) != "id").map(
    s => SmartMeterMetrics(s(0), s(1), s(2).toInt)
)
val telemetryDF = telemetryRDD.toDF()
telemetryDF.write.saveAsTable("SmartMeters")
```

6. Next, click the **Run** icon in the toolbar to execute this code and create the **SmartMeters** table.

7. The block is finished running when the In[*] changes to In[1]



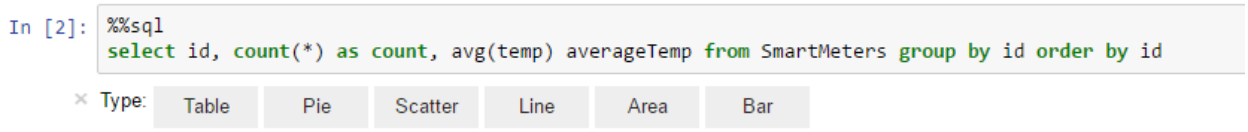
You may see the below message. You can proceed.

```
org.apache.spark.sql.AnalysisException: Table `SmartMeters` already exists.;
        at org.apache.spark.sql.DataFrameWriter.saveAsTable(DataFrameWriter.scala:232)
        at org.apache.spark.sql.DataFrameWriter.saveAsTable(DataFrameWriter.scala:221)
```

8. In the second cell, enter the following SQL query and run it.

```
%%sql
select id, count(*) as count, avg(temp) averageTemp from SmartMeters group by id order by
id
```

You will see a table like the following:



id	count	averageTemp
Device0	60	69.083333
Device1	59	68.677966
Device2	60	69.666667
Device3	59	69.949153
Device4	60	69.500000
Device5	60	69.766667
Device6	59	69.508475

9. Next, create a table that will summarize the telemetry collected using the previous query. In a new paragraph, try running the following query:

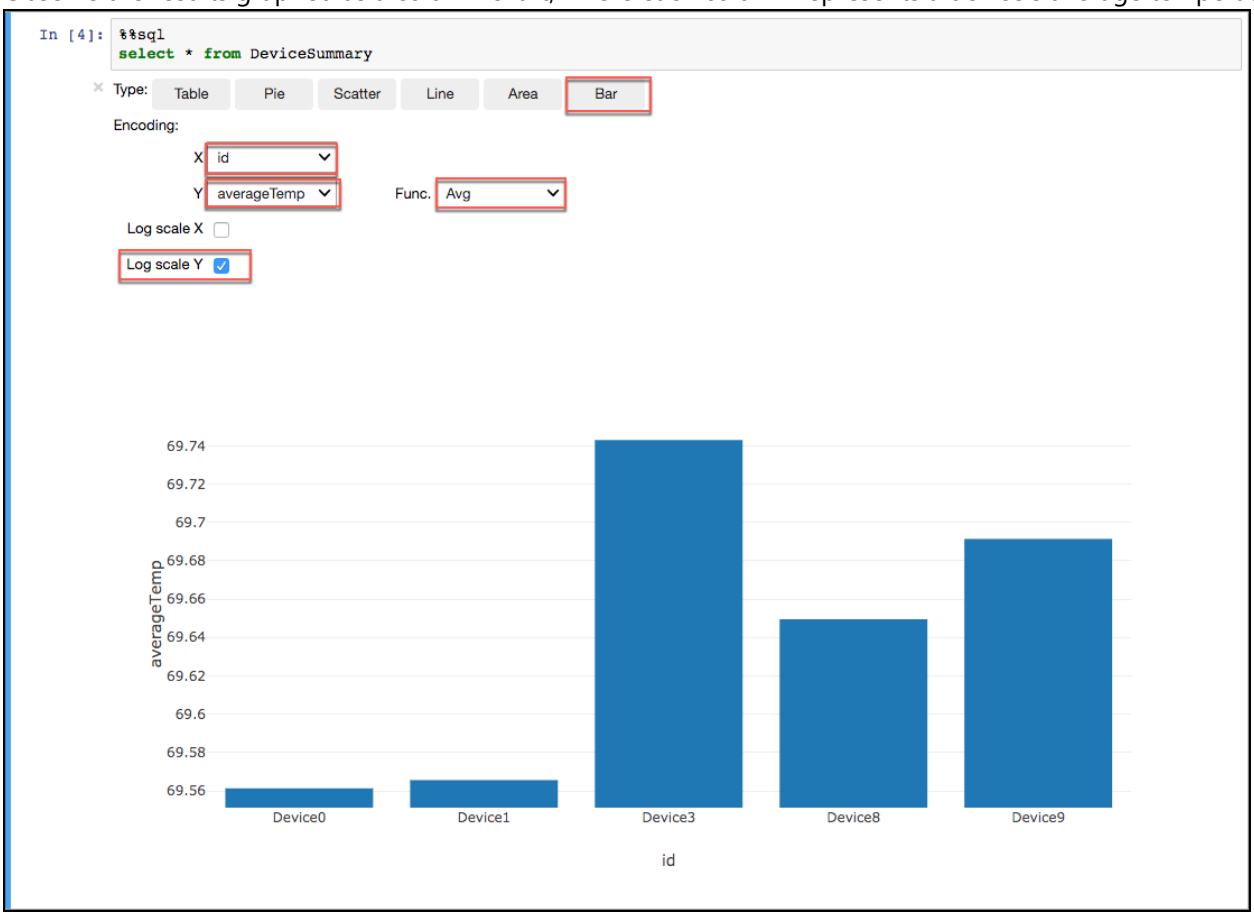
```
//query the table to create a summary result set
val summary = spark.sql("select id, count(*) as count, avg(temp) averageTemp from
SmartMeters group by id order by id")

//save the new pre-computed view
summary.write.saveAsTable("DeviceSummary")
```

10. Next, query from this summary table by executing the following query.

```
%%sql
select * from DeviceSummary
```

11. In the results, click the **Bar** button.
12. In the X dropdown, select id.
13. In the Y dropdown, select averageTemp.
14. In the Func dropdown, select Avg.
15. Check the box for Log scale Y.

16. Observe the results graphed as a column chart, where each column represents a device's average temperature.



Exercise 6: Reporting device outages with IoT Hub Operations Monitoring

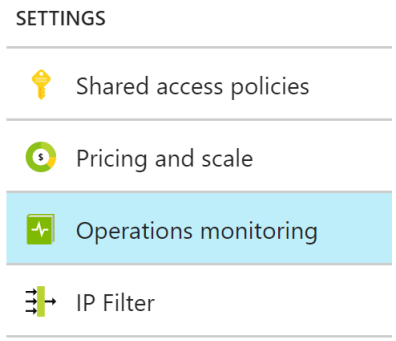
Duration: 20 minutes

Fabrikam would like to be alerted when devices disconnect and fail to reconnect after a period. Since they are already using PowerBI to visualize hot data, they would like to see a list of any of these devices in a report.

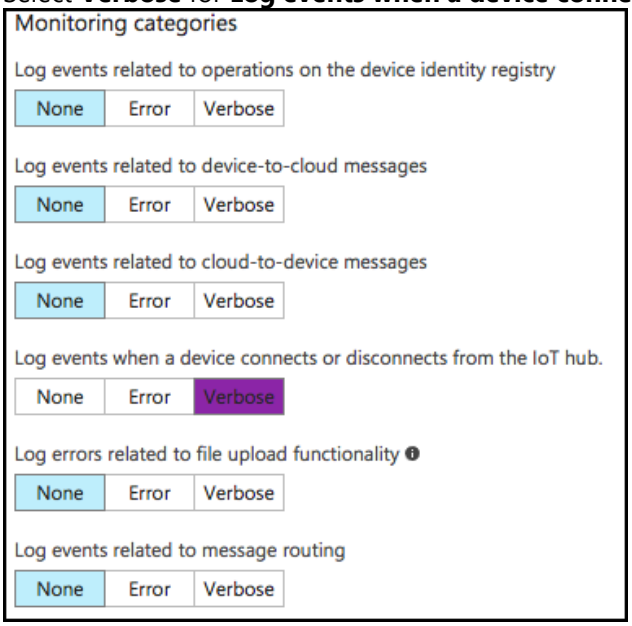
Task 1: Enable verbose connection monitoring on the IoT Hub

To keep track of device connects and disconnects, we first need to enable verbose connection monitoring.

1. In your browser, navigate to the **Azure Portal** (https://portal.azure.com).
2. Open the IoT Hub you provisioned earlier, **smart-meter**.
3. Under SETTINGS in the left-hand menu, click on **Operations monitoring**.



4. Select **Verbose** for **Log events when a device connects or disconnects from the IoT Hub**.

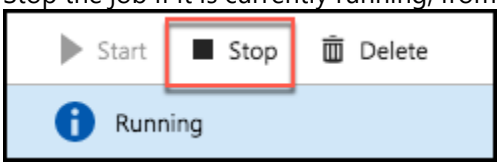


5. Click **Save**.

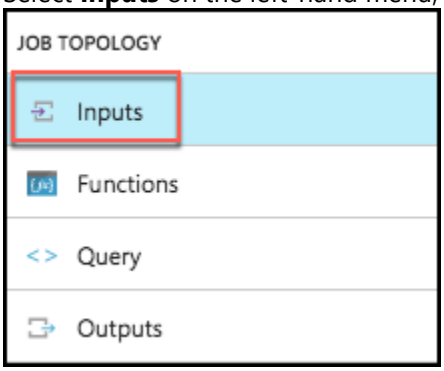
Task 2: Collect device connection telemetry with the hot path Stream Analytics job

Now that the device connections are being logged, update your hot path Stream Analytics job (the first one you created) with a new input that ingests device telemetry from Operations Monitoring. Next, create a query that joins all connected and disconnected events with a DATEDIFF function that only returns devices with a disconnect event, but no reconnect event within 120 seconds. Output the events to Power BI.

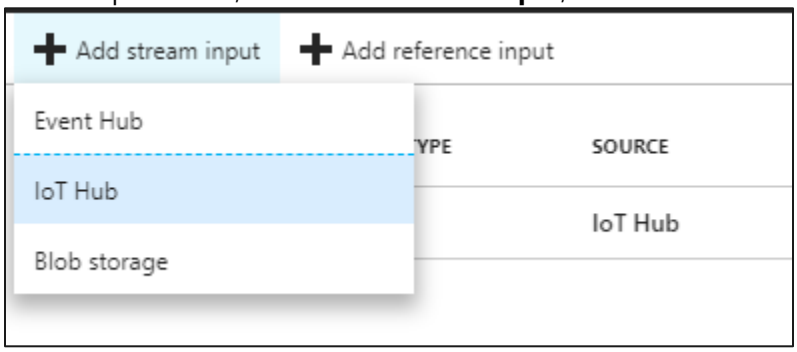
1. In your browser, navigate to the **Azure Portal** (https://portal.azure.com).
2. Open the **hot-stream** Stream Analytics job (the first one you created).
3. Stop the job if it is currently running, from the Overview blade, by selecting **Stop**, then **Yes** when prompted.



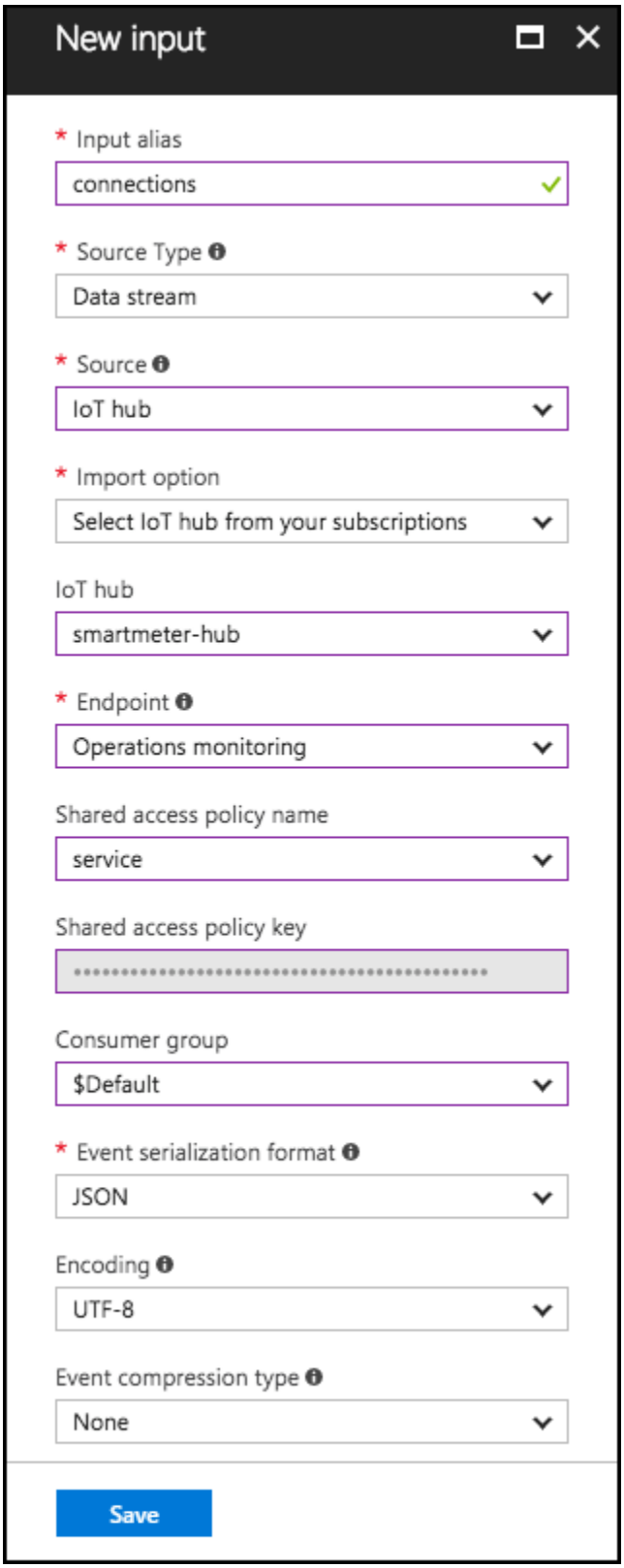
4. Select **Inputs** on the left-hand menu, under Job Topology.



5. On the Inputs blade, select **+Add stream input**, then select **IoT Hub** to add an input connected to your IoT Hub.

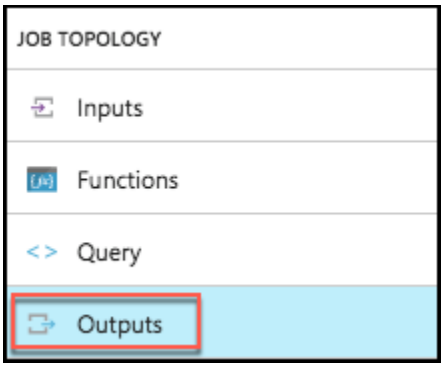


6. On the New Input blade, enter the following:
    a. Input Alias: Set the value to **connections**.
    b. Leave the **Select IoT hub from your subscriptions** option selected.
    c. IoT Hub: Select your existing IoT Hub, **smartmeter-hub**.
    d. Endpoint: Choose **Operations monitoring**.
    e. Shared Access Policy Name: Set to **Service**.
    f. Consumer Group: Leave as **$Default**.
    g. Event serialization format: Choose **JSON**.
    h. Encoding: Choose **UTF-8**.
    i. Event compression type: Leave set to **None**.

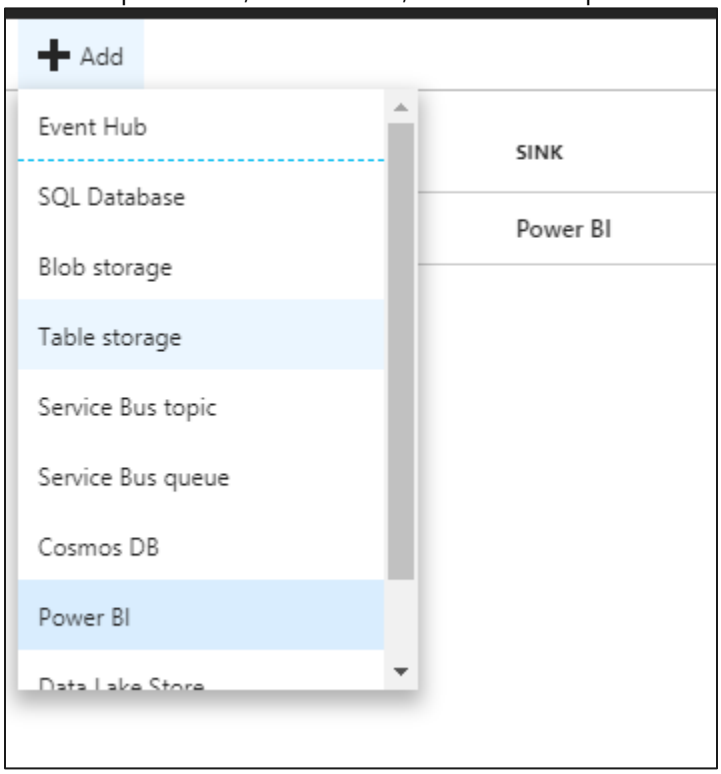


j. Select **Save**.

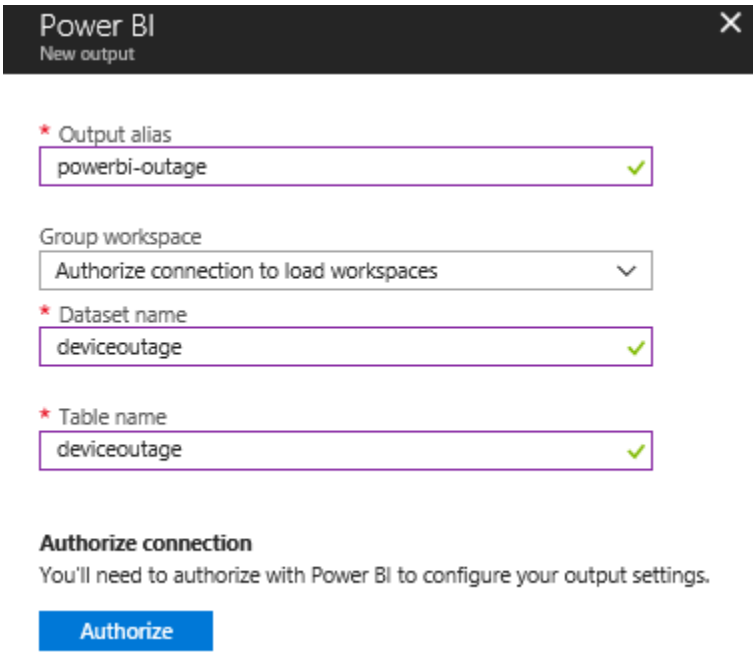
11. Now, select **Outputs** from the left-hand menu, under Job Topology.



12. In the Outputs blade, select **+Add**, to add the output destination for the query.

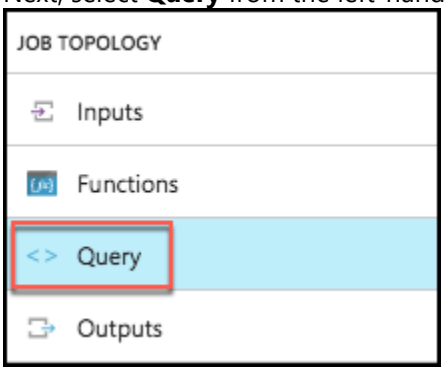


13. On the New output blade, enter the following:
    a. Set the **Output alias** to **powerbi-outage**.
    b. Set the **Group workspace** to **Authorize connection to load workspaces.**
    c. Dataset Name: Enter **deviceoutage**
    d. Table Name: Enter **deviceoutage**



e. Select **Authorize** under Authorize Connection.
Follow the on-screen prompts to log on to your Power BI account.
f. After authenticating, select **Save**.

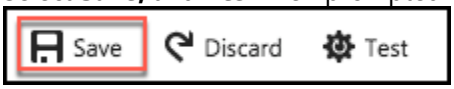
14. Next, select **Query** from the left-hand menu, under Job Topology.



15. We will replace the hot path query, which selects the averages of the temperatures into the PowerBI output, with queries that perform the following:
    a. Select device disconnection events.
    b. Select device connection events.
    c. Join these two streams together using the Stream Analytics DATEDIFF operation on the LEFT JOIN, and then filter out any records where there was a match. This gives us devices that had a disconnect event, but no corresponding connect event within 120 seconds. Output to the Service Bus.
    d. Execute the original hot path query.
16. Replace the existing query with the following, and click **Save** in the **command bar** at the top. (Be sure to substitute in your output aliases and input aliases):

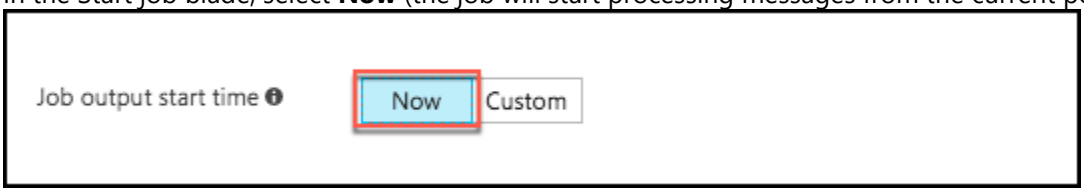
```
WITH
Disconnected AS (
SELECT *
FROM connections TIMESTAMP BY [Time]
WHERE OperationName = 'deviceDisconnect'
    AND Category = 'Connections'
),
Connected AS (
SELECT *
FROM connections TIMESTAMP BY [Time]
WHERE OperationName = 'deviceConnect'
    AND Category = 'Connections'
)

SELECT Disconnected.DeviceId, Disconnected.Time
INTO [powerbi-outage]
FROM Disconnected
LEFT JOIN Connected
    ON DATEDIFF(second, Disconnected, Connected) BETWEEN 0 AND 120
    AND Connected.deviceId = Disconnected.deviceId
WHERE Connected.DeviceId IS NULL;

SELECT AVG(temp) AS Average, id
INTO powerbi
FROM temps
GROUP BY TumblingWindow(minute, 5), id;
```

17. Select **Save,** and **Yes** when prompted with the confirmation.



18. Return to the Overview blade on your Stream Analytics job, and select **Start**.

19. In the Start job blade, select **Now** (the job will start processing messages from the current point in time onward).

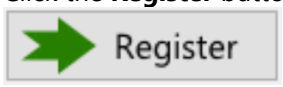


20. Select Start.
21. Allow your Stream Analytics Job a few minutes to start.

Task 3: Test the device outage notifications

Register and activate a few devices on the Smart Meter Simulator, then connect them. Deactivate them without reconnecting in order for them to show up in the device outage report we will create in the next task.

1. Run the Smart Meter Simulator from Visual Studio.
2. Click the **Register** button.

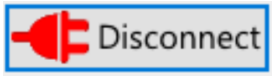


3. Click on 3 of the windows to highlight them.

4. Click the **Activate** button.

5. Click the **Connect** button.

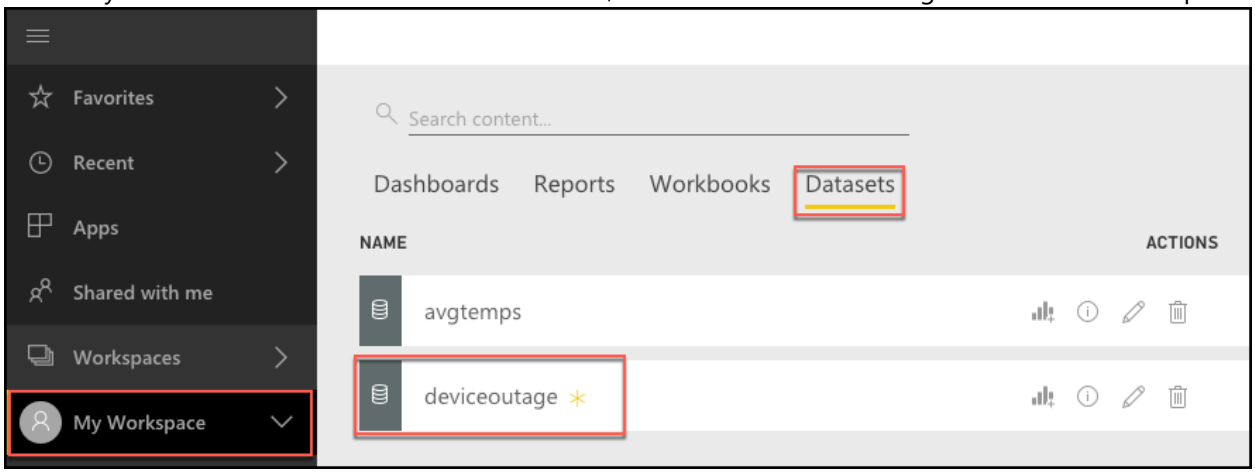
6. After a few seconds, click **Disconnect**.



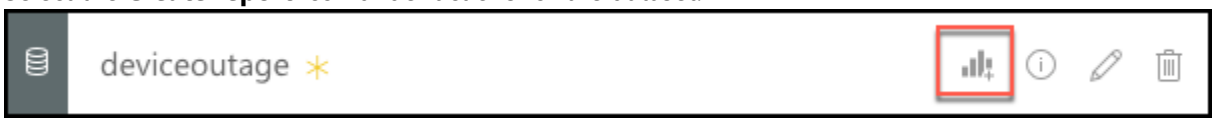
7. Click **Unregister**.

Task 4: Visualize disconnected devices with Power BI

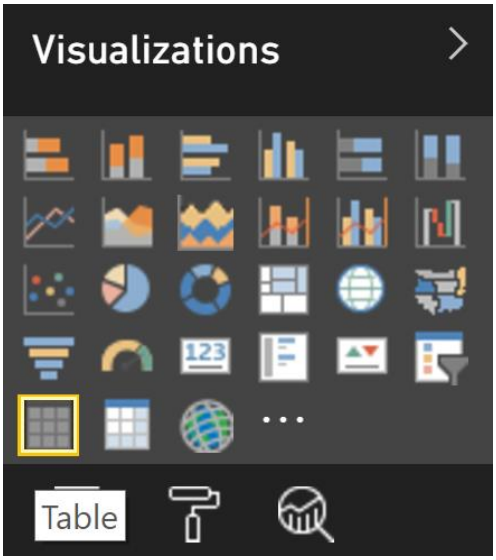
1. Log on to **Power BI** to see if data is being collected.
2. As done previously, select My Workspace on the left-hand menu, then select the Datasets tab. A new dataset should appear, named **deviceoutage**. (It is starred to indicate it is new) If you do not see the dataset, you may need to connect your devices on the Smart Meter Simulator, then disconnect and unregister them and wait up to 5 minutes.



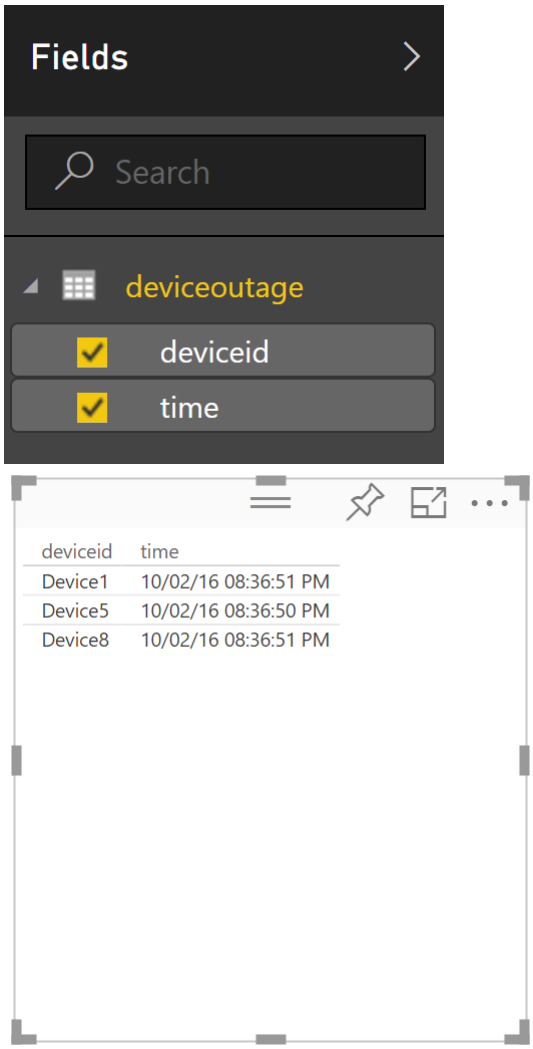
3. Select the **Create report** icon under actions for the dataset.



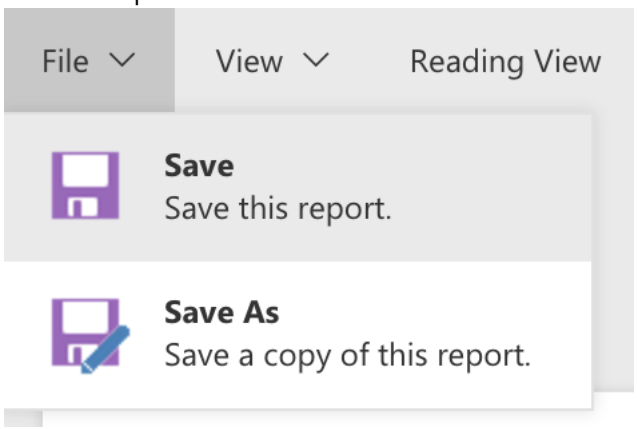
4. Add a **Table visualization**.



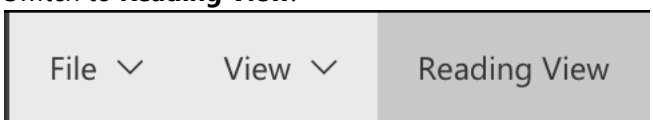
5. Select the **deviceid** and **time** fields, which will automatically be added to the table. You should see the Device Id of each of the devices you connected, and then disconnected for more than 2 minutes.



6. Save the report as **Disconnected Devices**.



7. Switch to **Reading View**.



8. Within the report, click the column headers to sort by device or date. You may run a few more tests with the Smart Meter Simulator and periodically refresh the report to see new devices.

deviceid	time
Device6	10/02/16 09:07:34 PM
Device1	10/02/16 09:07:33 PM
Device4	10/02/16 09:07:32 PM
Device5	10/02/16 09:07:32 PM
Device1	10/02/16 08:36:51 PM
Device8	10/02/16 08:36:51 PM
Device5	10/02/16 08:36:50 PM

After the hands-on lab

Duration: 10 minutes

In this exercise, attendees will deprovision any Azure resources that were created in support of the lab.

Task 1: Delete the resource group

1. Using the Azure portal, navigate to the Resource group you used throughout this hands-on lab by selecting **Resource groups** in the left menu.
2. Search for the name of your research group, and select it from the list.
3. Select **Delete** in the command bar, and confirm the deletion by re-typing the Resource group name, and selecting **Delete**.