**Configure Visual Studio Code and build DeviceHarness project**

1. Open a remote desktop session to your development VM.
2. In Visual Studio code, open the C:\source\IoTEdgeAndMlSample\DeviceHarness folder.
3. Validate that your environment is properly set up by triggering a build, **Ctrl** + **Shift** + **B**, or **Terminal** > **Run Build Task**.
4. You're prompted to select the build task to run. Select **Build**.
5. The build runs and outputs a success message.

**Connect to IoT Hub and run DeviceHarness**

Now that we have the project building, connect to your IoT hub to access the connection string and monitor the progress of the data generation.

**Sign in to Azure in Visual Studio Code**

1. Sign into your Azure subscription in Visual Studio Code by opening the command palette, Ctrl + Shift + P or **View** > **Command Palette**.
2. Search for the **Azure: Sign In** command. A browser window opens and prompts you for your credentials. When you're redirected to a success page, you can close the browser.

**Connect to your IoT hub and retrieve hub connection string**

1. In the bottom section of the Visual Studio Code explorer, select the **Azure IoT Hub** frame to expand it.
2. In the expanded frame, click on **Select IoT Hub**.
3. When prompted, select your Azure subscription and then your IoT hub.
4. Click the **...** to the right of **Azure IoT Hub** for more actions. Select **Copy IoT Hub connection string**.

**Run the DeviceHarness project**

1. Select **View** > **Terminal** to open the Visual Studio Code terminal.If you do not see a prompt, press Enter.
2. Enter dotnet run in the terminal.
3. When prompted for the IoT Hub Connection String, paste the connection string copied in the previous section.
4. In the **Azure IoT Hub devices** frame, click on the refresh button.
5. Note that devices are added to the IoT Hub and that the devices show up in green to indicate that data is being sent via that device. After the devices send messages to the IoT hub, they disconnect and appear blue.
6. You can view messages sent to the hub by right-clicking on any device and selecting **Start Monitoring Built-in Event Endpoint**. The messages will show in the output pane in Visual Studio Code.
7. Stop monitoring by clicking in the **Azure IoT Hub** output pane and choose **Stop Monitoring Built-in Event Endpoint**.
8. Let the application run to completion, which takes a few minutes.

**Check IoT Hub for activity**

The data sent by the DeviceHarness went to your IoT hub, where you can verify in the Azure portal.

1. Open the [Azure portal](https://portal.azure.com/) and navigate to the IoT hub created for this tutorial.
2. From the left pane menu, under **Monitoring**, select **Metrics**.
3. On the chart definition page, click the **Metric** drop down, scroll down the list, and select **Routing: data delivered to storage**. The chart should show the spike of when the data was routed to storage.

**Validate data in Azure Storage**

The data we just sent to your IoT hub was routed to the storage container that we created in the previous article. Let’s look at the data in our storage account.

1. In the Azure portal, navigate to your storage account.
2. From the storage account navigator, select **Storage Explorer (preview)**.
3. In the storage explorer, select **Blob Containers** then devicedata.
4. In the content pane, click on the folder for the name of the IoT hub, followed by year, month, day, and hour. You will see several folders representing the minutes when the data was written.
5. Click into one of those folders to find data files labeled 00 and 01 corresponding to the partition.
6. The files are written in [Avro](https://avro.apache.org/) format. Double-click on one of these files to open another browser tab and partially render the data. If you're prompted to open the file in a program, you can choose VS Code and it will render correctly.