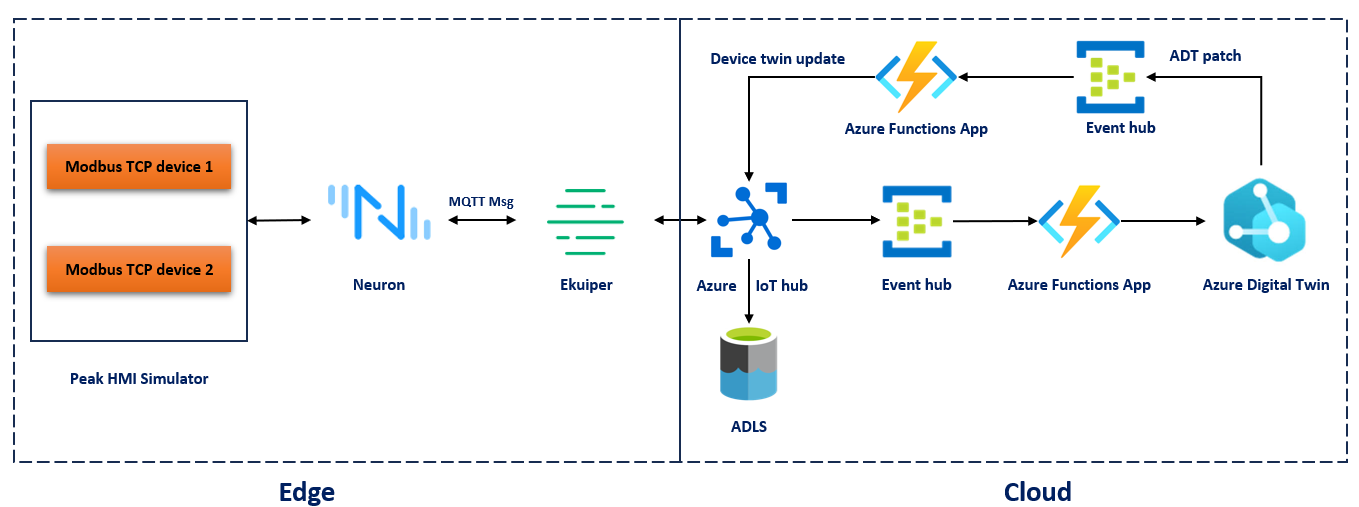
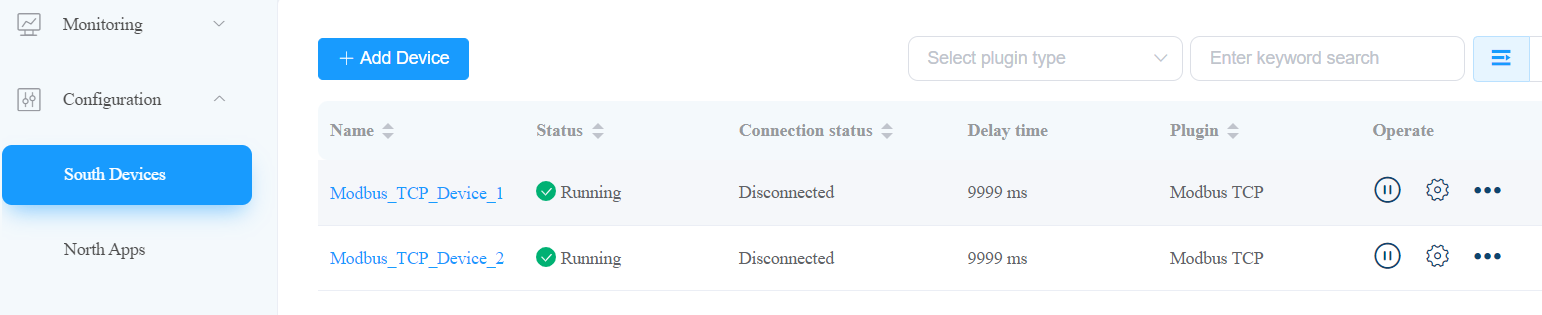
**Documentation**

The architecture of the IIoT telemetry system to send industrial devices data to Azure Digital Twin and vice versa.

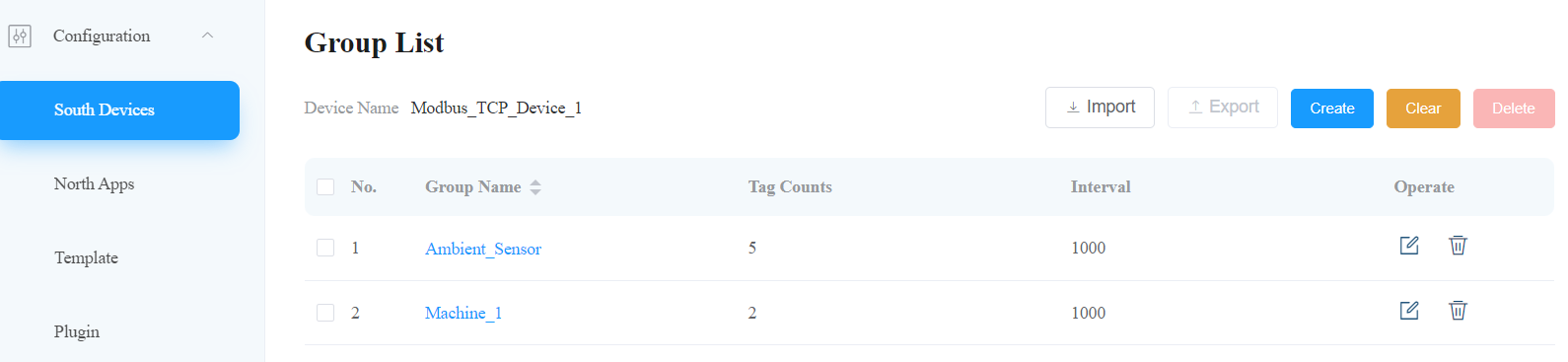


1. **Set up PeakHMI Modbus TCP simulator**
   1. Download PeakHMI Modbus TCP simulator from this [link](https://www.hmisys.com/downloads/PeakHMISlaveSimulatorInstall.exe)
   2. After installing the simulator go to windows →Register data, in the new window select the device ID, each Modbus TCP device will have a unique address, The devices that are connected to Modbus devices will be identified by the device address.
   3. After selecting the device ID, the data monitor window will open. In general the sensors connected to the Modbus devices will be read periodically and stored in a holding register. To simulate the sensor value click the row against the address under the float column. Enter the value and click ok. Float data type uses 16 bit addressing, two holding registers will be used for storing float values. Eg, address 400001 and 400002 will be used for a single variable. To add data to the second sensor use address 400003.
   4. The attribute type “read” will be used to read values from the holding register and “write” will be used to write values into the holding register. Write attributes useful in scenarios where we need to control the device from upstream services/applications.
   5. To simulate data for a second Modbus TCP device, go to windows → register data → select 2 then repeat the above process to set values. You can notice two device monitor windows will have corresponding ID’s 1 and 2.
2. **Setup Neuron**
   1. Neuron can be installed on linux machines or linux VM or WSL. While installing it one WSL make sure that “[Distrod](https://github.com/nullpo-head/wsl-distrod)” and [Docker](https://nickjanetakis.com/blog/install-docker-in-wsl-2-without-docker-desktop) is installed before the installation. Use install option 2 from the distrod link and ignore step 3 in docker link.
   2. Install the Neuron application from this [link](https://neugates.io/docs/en/latest/installation/neuron/docker.html#get-the-image). Neuron can be installed as a stand alone linux application or container application. We recommend installing the Neuron container application.
   3. After the installation starts the neuron application using docker command “docker start neuron” if application is not started.
   4. Open any web browser and type “[http://IPaddress:7000](http://ipaddress:7000)” to start the web UI of Neuron. Ip Address is the IP address of the machine where Neuron is installed. In WSL type “ifconfig” to find the IP address.
   5. The default login credentials are username: admin, password:0000
   6. Add the south bound device (Modbus TCP slave) using this [link](https://neugates.io/docs/en/latest/configuration/south-devices/south-devices.html). Don't install the Modbus slave tool mentioned in the tutorial. Only follow the tutorials on Neuron side. The IP address mentioned in the “device configuration” is the Ip address of the PeakHMI Modbus TCP simulator. The first number in the Tag address (1!400001) is the Modbus device ID and 400001 is the holding register address where data is stored.

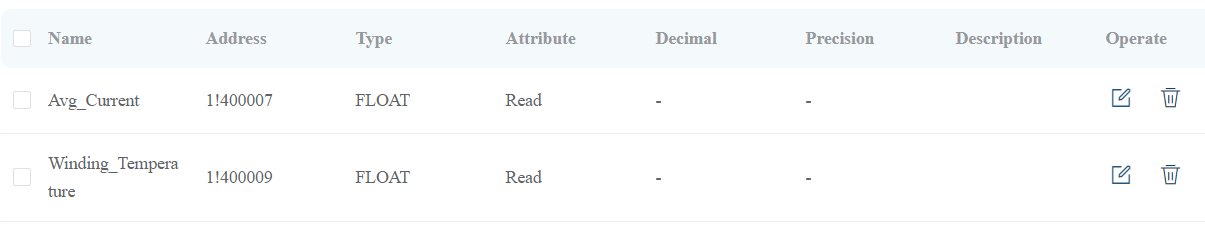
The following image shows two devices are added in Neuron



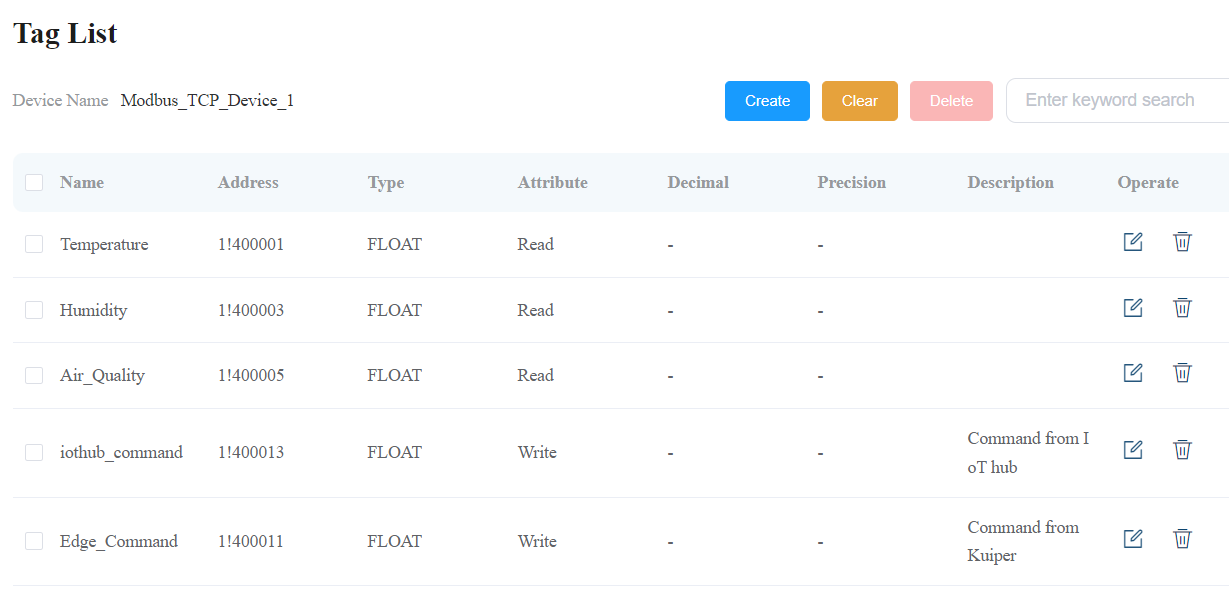
Two groups are created inside the Modbus\_TCP\_Device\_1



Tag list for the Machine\_1 group

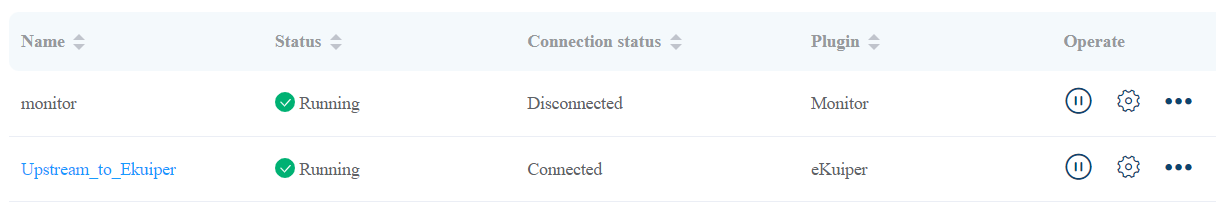


Tag list for the ambient sensor group

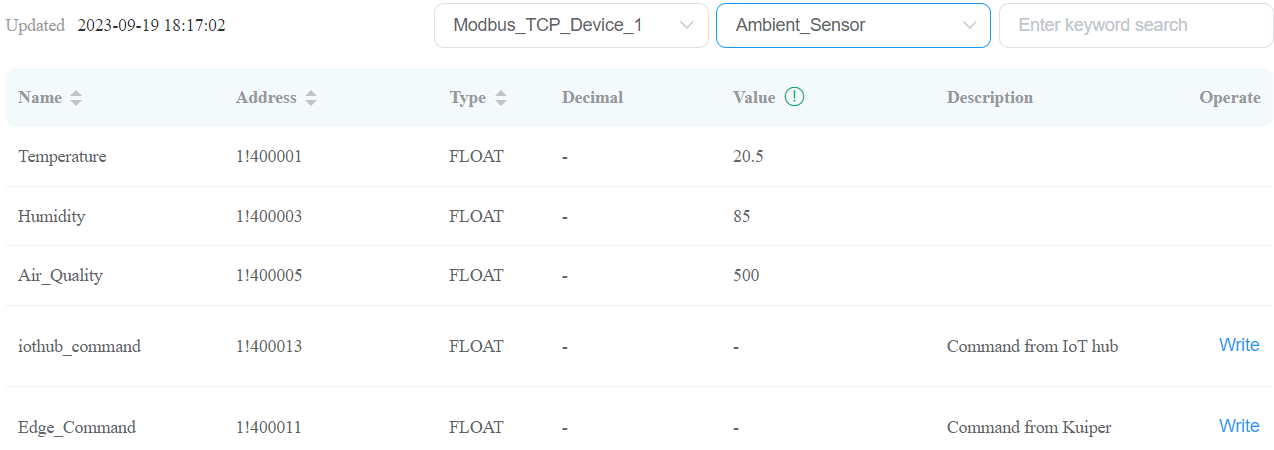


* 1. We send the telemetry data to upstream application Ekuiper for edge analytics. Add the Northbound application using this [link](https://neugates.io/docs/en/latest/configuration/north-apps/ekuiper/overview.html). The IP address mentioned in the tutorial is the Neuron container IP address. It can be obtained by using “docker inspect neuron | grep IPAddress” from WSL.

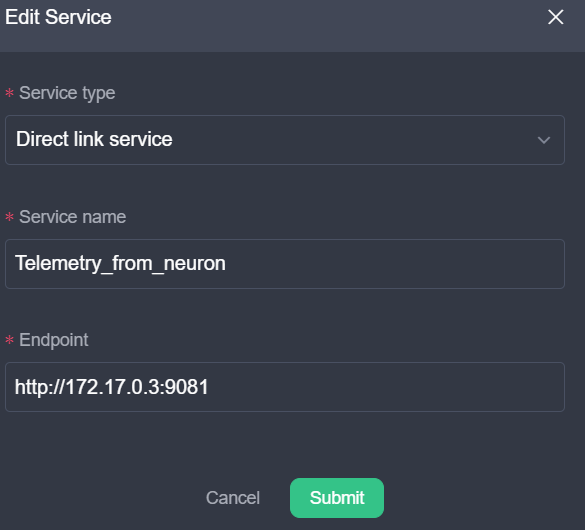
North bound application



* 1. You can use the data monitor option to check the telemetry data by providing the south device name and group name.



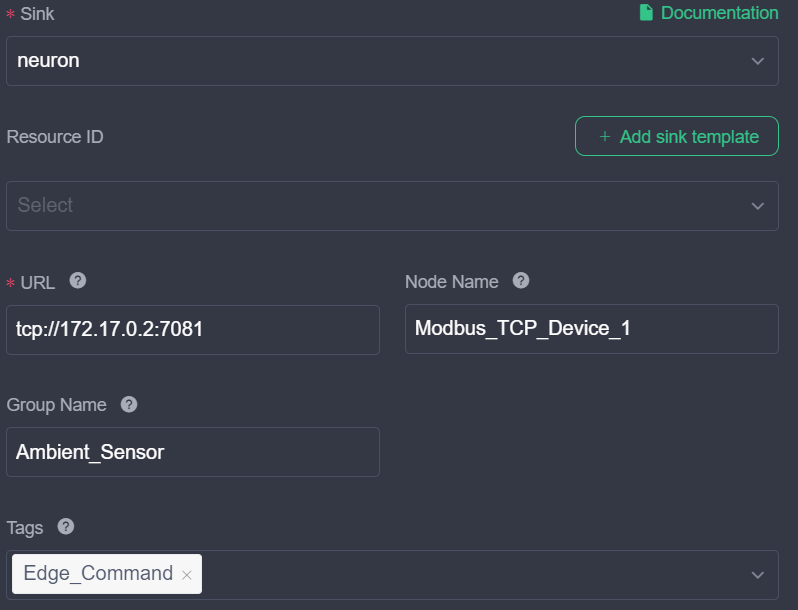
1. **Setup Ekuiper**
   1. Similar to Neuron, Ekuiper can also be installed as a standalone or container application on linux VM or WSL.
   2. We recommend installing Ekuiper as a container application using this [link](https://ekuiper.org/docs/en/latest/installation.html#running-ekuiper-in-docker). Install Ekuiper with a management console to set up web based user configuration and rule creation.
   3. Make sure the container is up and running by using the “docker ps” command from WSL.
   4. Open any web browser and type “[http://Ipaddress:9082](http://ipaddress:7000)” to start the web UI of Ekuiper manager. Ip Address is the IP address of the machine where Neuron is installed. In WSL type “ifconfig” to find the IP address.
   5. Use this tutorial link to add the service to Ekuiper manager. Skip to “getting started” if Ekuiper installed previously using docker compose, otherwise install the Ekuiper manager. IP address mentioned in the add service is the IP address of the Ekuiper container, it can be obtained using “docker inspect kuiper | grep IPAddress'' from WSL. Note that creating the stream section is optional.



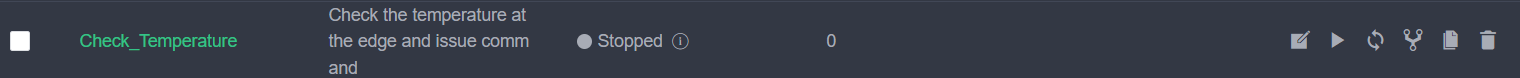
* 1. Now we have to receive the telemetry stream from downstream application Neuron. To do that, first we have to create a service. Click the create stream button on the right side of the window, now input the stream name and select the stream type as “neuron” from the drop down menu and leave the rest of the field default and click submit.
  2. Now the stream is created, let’s create some rules to process the stream. In Ekuiper, rules can be created using SQL query or using a graph. First we create a rule using SQL query, click the Rules tap and then click Create rule. Give rule ID, rule name. In the SQL query box we have to enter the queries. For example the following query will check the average temperature for every 20 seconds from the stream “Test\_stream'' with group name “Ambient\_Sensor” and if it exceeds 40 then Edge\_Command will be set as “1”.

select 1 as Edge\_Command from Test\_stream where group\_name="Ambient\_Sensor" group by tumblingwindow(ss,20) having avg(values.Temperature)>40

* 1. Next we have to set the action, The command has to be sent to the neuron, click the Add button and select sink type as “neuron”. Fill the URL, node, group and tag name. Note that the tag attribute should be “write” in order to write value to the holding registry of Modbus TCP device. Click the submit button of the action window and rule window.

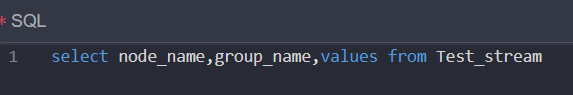


* 1. If the rule is not running then click the play button, if there is no error the rule will run and perform the task. You can also see the output of the rule by clicking on Running status.



* 1. Similarly we have to create a rule to send data to the IoT hub. Repeat the above steps and while adding action select sink type as MQTT and fill the required details.

SQL query



* 1. MQTT Configuration page

Use this [link](https://learn.microsoft.com/en-us/azure/iot/iot-mqtt-connect-to-iot-hub) to connect devices to Azure IoT hubs using MQTT protocol.

MQTT broker address : ssl://ESP32Experiment.azure-devices.net:8883

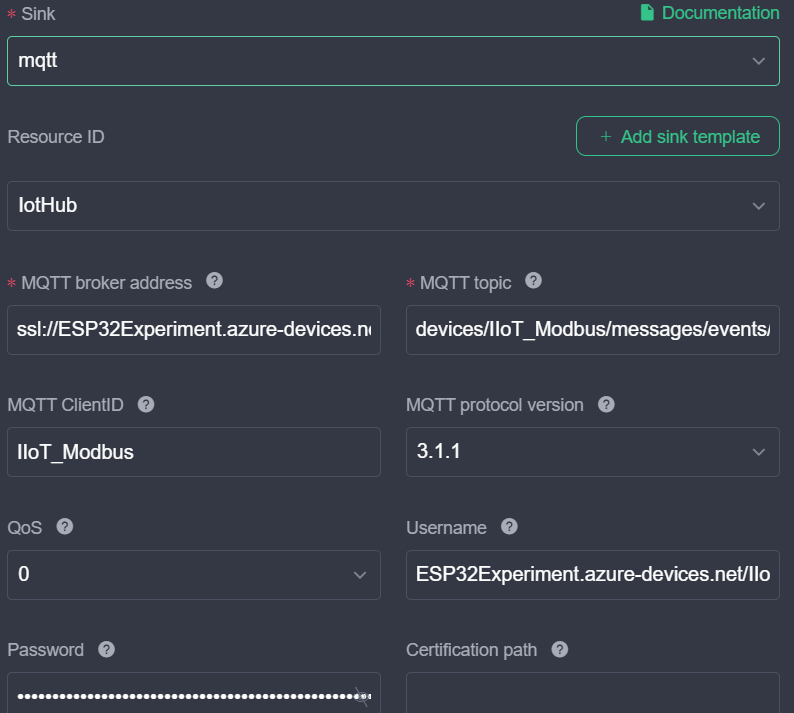
MQTT topic : devices/IIoT\_Modbus/messages/events/$.ct=application%2Fjson%3Bcharset%3Dutf-8

MQTT Client ID is the device name created in Azure IoT hub.

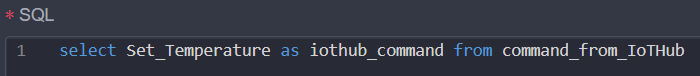
User name : ESP32Experiment.azure-devices.net/IIoT\_Modbus/?api-version=2021-04-12

Password will be the SAS token, It has to be generated using Azure extension for VScode. Refer to the above link.

Note that “ESP32Experiment” is the IoT hub name and “IIoT\_Modbus” is the device name created in IoT hub.



* 1. We have to create another stream to receive the device twin patch from the IoT hub. Repeat the step “f”. Select stream type as MQTT then enter the following MQTT topic “$iothub/twin/PATCH/properties/desired/#”. Then click add configuration key and use the above mentioned MQTT credentials and then submit.
  2. Create a rule to receive the command from the new stream we have created. The following query receives the “Set\_Temperature” command from the stream “command\_from\_IoTHub”. Note that Set\_Temperaure is renamed as iothub\_command which is the tag name. Otherwise it will not be executed. Even Though we mention this in the neuron sink configuration page, it should also be set here. Next add a new action and select sink type as neuron and fill the details.



1. **Azure IoT hub setup**
2. Use this [link](https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal) to set up an IoT hub and add a new device to the hub. Note that the name of the device created in the IoT hub for the demo is “IIoT\_Modbus”.
3. Use this [link](https://learn.microsoft.com/en-us/azure/iot-hub/how-to-routing-portal?tabs=storage) to send telemetry to the azure blob storage.
4. Create an event subscription in the IoT hub for the telemetry events for the device we have created. Select event type as “Device Telemetry” and end point type as azure function while creating event subscription. In the filter tap enable subject filtering and set the subject name as device name, in our case it is “IIoT\_Modbus”. Note that Azure function should be created and ready before creating event subscription. We have to provide the end point name while creating the event.
5. **Set up Azure function**
6. Use this [link](https://learn.microsoft.com/en-us/azure/azure-functions/functions-create-function-app-portal) to create an azure function app from azure portal.
7. Use this [link](https://learn.microsoft.com/en-us/azure/azure-functions/functions-develop-vs-code?tabs=node-v3%2Cpython-v2%2Cin-process&pivots=programming-language-python) to set up the azure function extension in VS code.
8. We use python to create an azure function in this tutorial. Install [python](https://docs.python.org/3.9/using/windows.html) or [anaconda](https://docs.anaconda.com/free/anaconda/install/windows/), libraries such as [Azure IoT core](https://pypi.org/project/azure-iot-hub/), [Identity](https://pypi.org/project/azure-identity/), [Azure digital twin](https://pypi.org/project/azure-digitaltwins-core/), [storage](https://pypi.org/project/azure-storage/) and [Azure IoT hub](https://pypi.org/project/azure-iot-hub/) should be installed.
9. Create a new azure function using this repository, debug and deploy.
10. Note that there are two functions, “modbus\_to\_adt” for receiving the telemetry events from IoT hub and updating the digital twin. “Twin\_iothub\_update” function receives the digital twin patch (in this demo Set\_Temperature property) and updates the device twin of the device created in the IoT hub.
11. **Set up Azure Digital twin**
    1. Create a Azure digital twin from Azure portal using this [link](https://learn.microsoft.com/en-us/azure/digital-twins/how-to-set-up-instance-portal)
    2. Create Digital twin models using this [link](https://learn.microsoft.com/en-us/azure/digital-twins/concepts-models). You can find the models for this demo from this link. You can manually upload the models to the digital twin explorer using this [link](https://learn.microsoft.com/en-us/azure/digital-twins/how-to-use-azure-digital-twins-explorer).
    3. Create a digital twin for the models and relationship between them. You can use the python script provided in the repo or you can do it manually on the digital twin explorer.