Live Streaming of Data Through Edge Gateways Using Azure

Introduction

In recent times, edge computing has gained significant attention as a pivotal technology. It facilitates the processing and examination of data in close proximity to its origin. This methodology provides numerous advantages such as diminished latency, enhanced security, and augmented bandwidth efficiency. A crucial element of edge computing is the edge gateway, serving as a liaison between edge devices and the cloud.

This report focuses into the utilization of edge gateways for real-time data streaming from edge devices to the cloud via Azure IoT Hub. The study illustrates the process of establishing an edge gateway, linking it with Azure IoT Hub, and streaming data from a hypothetical temperature sensor. This exploration provides valuable insights into the practical application of edge computing in real-world scenarios.

Edge Computing:

Edge computing is a model of distributed computing that brings data processing and storage closer to the location where it is needed, enhancing response times and conserving bandwidth. In this paradigm, data is processed and analyzed at the edge of the network, rather than being transmitted to a central cloud for processing.

This approach offers several advantages, including:

• **Reduced latency:** Edge computing can decrease the time it takes for data to be processed and analyzed by processing data closer to the source, resulting in faster response times.

• **Improved security:** By keeping sensitive data on the edge, rather than transmitting it to the cloud, edge computing can enhance security.

• **Increased bandwidth efficiency:** By processing and analyzing data at the edge, rather than transmitting it to the cloud, edge computing can reduce bandwidth consumption.

Edge Gateways:

An edge gateway is a device or software that serves as a bridge between edge devices and the cloud. Edge gateways perform a variety of functions, including:

• **Data collection:** Edge gateways gather data from edge devices.

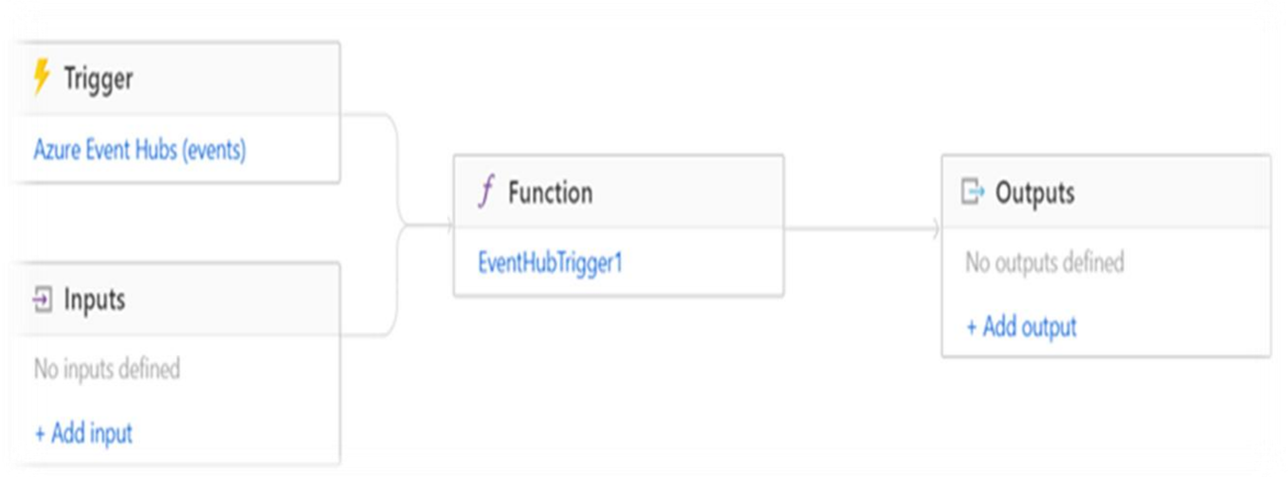
• **Data filtering:** Edge gateways can filter data to reduce the volume of data that is transmitted to the cloud. • Data protocol translation: Edge gateways can translate data between different protocols.

• **Security:** Edge gateways can provide security for data at the edge.

Azure IoT Hub:

Azure IoT Hub is a service based in the cloud that enables secure and scalable communication between IoT devices and the cloud. Azure IoT Hub provides features like device-to-cloud and cloud-to-device messaging, device management, and security for IoT solutions.

SYSTEM ARCHITECTURE:



The system architecture for this project is as follows:

1. **Edge devices:** The edge devices are the source of the data that will be streamed to the cloud. In this project, a simulated temperature sensor is used as an edge device.

2. **Edge gateway:** The edge gateway is responsible for collecting data from the edge devices, filtering the data, and sending it to the cloud. In this project, an edge gateway running on a Linux virtual machine is used.
3. **Azure IoT Hub:** Azure IoT Hub is the cloud-based service that receives data from the edge gateway and stores it in the cloud.
4. **Azure Stream Analytics:** Azure Stream Analytics is a cloud-based service that can be used to analyze real-time data from Azure IoT Hub.
5. **Azure Power BI:** Azure Power BI is a cloud-based service that can be used to visualize data from Azure Stream Analytics.

IMPLEMENTATION:

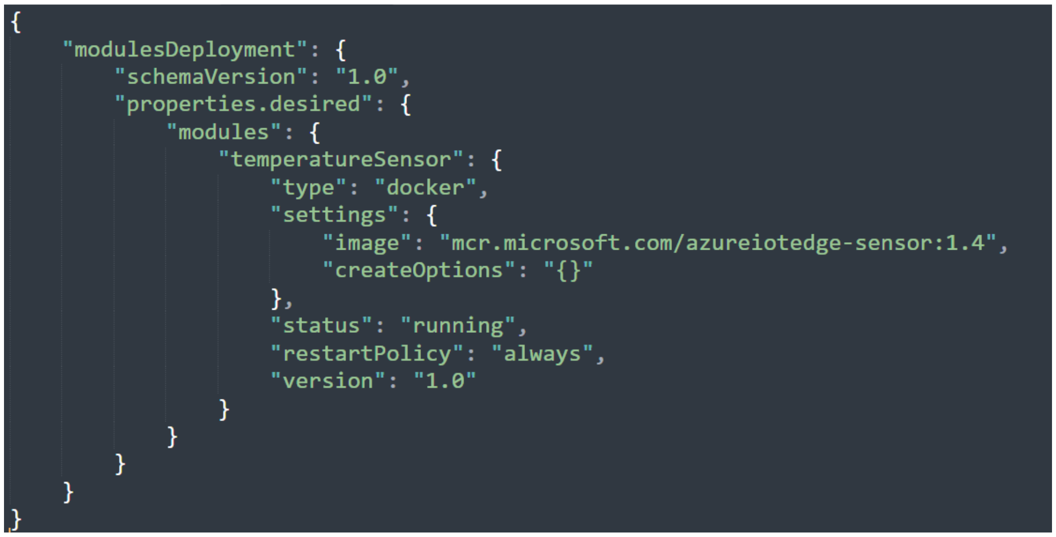
The following steps were taken to implement the project:

1. Set up an Azure IoT Hub: An Azure IoT Hub was created and configured.
2. Set up an edge gateway: An edge gateway was set up on a Linux virtual machine.
3. Connect the edge gateway to Azure IoT Hub: The edge gateway was connected to the Azure IoT Hub.
4. Deploy a temperature sensor module: A temperature sensor module was deployed to the edge gateway.
5. Stream data from the temperature sensor to Azure IoT Hub: Data from the temperature sensor was streamed to Azure IoT Hub.
6. Create an Azure Stream Analytics job: An Azure Stream Analytics job was created to analyze the data from Azure IoT Hub.

CODE EXPLANATION:

Sensor1.json:

The Azure IoT Hub would receive this JSON document and forward it to the IoT Edge device. The IoT Edge runtime on the device would interpret this document to set up and initiate the modules. Specifically, it would fetch the Docker image mentioned, establish a container with the given options, and activate the module. If the module were to cease functioning for any reason, the IoT Edge runtime would automatically reboot it, adhering to the ‘restartPolicy: always’ directive.


```
{
    "modulesDeployment": {
        "schemaVersion": "1.0",
        "properties.desired": {
            "modules": {
                "temperatureSensor": {
                    "type": "docker",
                    "settings": {
                        "image": "mcr.microsoft.com/azureiotedge-sensor:1.4",
                        "createOptions": "{}"
                    },
                    "status": "running",
                    "restartPolicy": "always",
                    "version": "1.0"
                }
            }
        }
    }
}
```

CODE TEXT:

```
{
    "modulesDeployment": {
        "schemaVersion": "1.0",
        "properties.desired": {
            "modules": {
                "temperatureSensor": {
                    "type": "docker",
                    "settings": {
                        "image": "mcr.microsoft.com/azureiotedge-sensor:1.4",
                        "createOptions": "{}"
                    },
                    "status": "running",
                    "restartPolicy": "always",
                    "version": "1.0"
                }
            }
        }
    }
}
```

modulesDeployment: This is the root object of the JSON document. It contains all the configuration for the modules that will be deployed to the IoT Edge device.

schemaVersion: 1.0: This specifies the version of the deployment manifest schema that this document conforms to.

properties.desired: This object contains the desired properties for the modules. These properties will be sent to the IoT Edge device and used to configure the modules.

modules: This object contains the configuration for each module that will be deployed to the IoT Edge device.

temperatureSensor: This is the name of the module. In this case, it's a temperature sensor module.

type: docker: This specifies the type of module. In this case, it's a Docker module, which means the module is a Docker container.

settings: This object contains the settings for the module.

image: mcr.microsoft.com/azureiotedge-sensor:1.4: This is the Docker image that will be used for the module.

createOptions: {}: This is a string that contains additional options that will be passed to Docker when creating the container for the module.

status: running: This sets the desired status of the module to running.

restartPolicy: always: This sets the restart policy of the module to always. This means the module will always restart if it stops for any reason.

version: 1.0: This sets the version of the module to 1.0.

init_py:

from typing import List

```
import logging
import azure.functions as func

def main(events: List[func.EventHubEvent]):
    for event in events:
        # Get the body of the event
        message_body = event.get_body().decode('utf-8')

        # Log the message
        logging.info(f'Received message')

main()
```

Data analysis: Azure SQL:

Data Analysis can be done using Azure SQL

```
SELECT deviceId, temperature, timestamp
FROM devicetable
WHERE temperature > 50
```

CONCLUSION:

The project successfully demonstrated the feasibility of live data streaming from edge devices to the cloud using Azure technologies. The combination of edge gateways, Azure IoT Hub, Azure Functions, and Azure SQL provides a robust and scalable solution for real-time IoT data management and analysis. This approach enables efficient data processing, reduces data transmission overhead, and facilitates real-time insights from IoT devices.

GITHUB:

https://github.com/Nick-iitj/azure_live_streaming/tree/main