IoT Temperature Monitoring System

Overview

My final Task project is **IoT Temperature Monitoring System** which demonstrates an Edge-to-UI architecture, where a web application that collects, stores, and visualizes temperature data from IoT devices. The system includes a temperature sensor, a backend API to handle data ingestion and querying, a database for storage, and a user-friendly interface to display real-time temperature readings and historical data trends. To more detailed, Raspberry Pi Pico W acts as the edge device and communicates with a Node.js Express.js backend (cloud) using HTTP requests. The Pico W collects data (e.g., sensor readings) and sends it to the cloud via HTTP GET requests. The cloud backend receives this data through a REST API endpoint that accepts a query parameter value, then Grafana is used, which is an open-source tool that provides an interactive dashboard for real-time and historical data visualization.

Features

- **IoT Sensor Simulation**: Sends temperature readings to the backend API periodically.
- **API Endpoint**: Receives and stores temperature data in an InfluxDB database.
- **Data Visualization**: Displays current temperature and historical trends in a simple web interface.
- **Scalable Backend**: Built with Node.js and Express.js for efficient data handling.
- **Frontend Integration**: Uses Grafana dashboard to provide interactive charts.

Technology Stack

- **Frontend**:
    - HTML, CSS, JavaScript

        - Grafana for data visualization
- **Backend**:
        - Node.js with Express.js
        - RESTful API design
        - InfluxDB for time-series data storage
- **DevOps**:
        - Nginx for reverse proxy and (CORS management for testing locally)
        - Systemd for service management
        - Git for version control

Installation

Prerequisites

- Node.js (v2x recommended)
- InfluxDB
- Git
- Nginx

---

1. Edge Device: Raspberry Pi Pico W + NTC Temperature Sensor

This is a sample implementation using Wokwi to simulate the edge devices. The official Edge is developed under the directory **Edge**. The technologies used are the same as used in **Edge**.

The **NTC Temperature Sensor** is a Wokwi Analog temperature sensor which has a negative temperature coefficient (NTC).

The **Raspberry Pi Pico W** is connected to a Wi-Fi network and sends HTTP requests to the cloud using the `urequests` module in MicroPython.

Purpose of this sample is to demonstrate how data can be collected from edge to the cloud.

Project codebase

Code files:

- `main.py` - Main program file
- `diagram.json` - Wiring schematic to the MCU (Wokwi)
- `readme.md` - Description of this sample

---

2. Resource Allocation and Benefits

Operating Costs

- **Azure Student VPS**: Free credits under Azure Student Plan.
- **InfluxDB** (Open Source): Open-source for time-series data storage.
- **Node.js**: Free and open-source runtime for backend services.
- **Nginx**: Free and lightweight HTTP server for reverse proxying.
- **Grafana** (Open Source): Provides interactive and customizable dashboards. Free tier supports all core functionalities.

Benefits of Selected Services

- **Azure Student VPS**: Ensures high availability with reliable infrastructure. No upfront cost for students under the Azure for Students plan.
- **InfluxDB (v2.7.10)**: Optimized for high-speed time-series data ingestion, enabling seamless storage and retrieval of IoT temperature data.
- **Node.js (v23)**: Asynchronous, non-blocking runtime ensures efficient data processing, perfect for handling the real-time demands of IoT systems.
- **Nginx (v1.24.0)**: Secure and scalable HTTP server for managing web traffic and ensuring smooth frontend-backend communication.

---

3. Cloud Environment Setup

Azure VPS Configuration

The Azure VPS is equipped with:

- **Backend Runtime**: Node.js v23.1.0 for efficient backend processing.
- **Database**: InfluxDB v2.7.11 for fast and reliable time-series data storage.
- **HTTP Server**: Nginx v1.24.0 for reverse proxying and load balancing.
- **Other Tools**:
    - Grafana: v11.4.0 for data visualization.
    - Systemd: (255.4-1ubuntu8.4) for service management to ensure 24/7 availability.
    - Curl: v8.5.0 for HTTP Requests and Testing APIs

Advantages

- **Latest Software Versions**: Ensures optimal performance and security.
- **Global Accessibility**: Public-facing services are available on standard HTTP (port 80) and HTTPS (port 443) protocols.
- **Seamless Integration**: All services are pre-configured for smooth interaction within the IoT pipeline.

---

4. Cloud Service Configuration

24/7 Operations

- Services are configured with **systemd** to ensure automatic startup and recovery in case of failures.
- Resource allocation on Azure ensures no interruptions even under high workloads.
- Here is how I made my db_api-service.service as 24/7 service operations:
    i. Accessing systemd Configuration at : /etc/systemd/system/
    ii. Create a Service File: sudo nano /etc/systemd/system/db_api-service.service and add below configuration:
    [Unit]

Description=Backend system for data delivery in IoT Pipeline
After=network.target
[Service]
ExecStart=/home/azureuser/.nvm/versions/node/v23.1.0/bin/node /opt/db_api/src/server.mjs
WorkingDirectory=/opt/db_api
Restart=always
RestartSec=10
User=db_api_user
Group=db_api_group
EnvironmentFile=/opt/db_api/.env
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=db_api-service
[Install]
WantedBy=multi-user.target

- Verify which services are running:
  `systemctl list-units --type=service`
  `cat /etc/systemd/system/db_api-service.service`

Connectivity and Firewall Configuration

- **Public Ports**: Connectivity services listens at the public ports e.g., 80 & 443
  - Port 80 (HTTP) and Port 443 (HTTPS) are open for frontend and API communication.
  - I did configuration for Nginx to on Ports 80 and 443 (I used Certbot to get SSL certificate for my domain at myiot.northeurope.cloudapp.azure.com), see at :
    `cat /etc/nginx/sites-available/default`
  - Verify that Nginx is Listening on Ports 80 and 443 by running:
    `sudo netstat -tuln | grep ':80\|:443'`
- **Firewall Rules & Routing**: Firewall allows and router routes traffic properly to the public ports properly. Nginx routes traffic appropriately to the backend or static web interface.
  - Configured to allow inbound traffic on necessary ports.
  - Here are what I did:

        - Allow HTTP (port 80)
          `sudo ufw allow 80`
        - Allow HTTPS (port 443)
          `sudo ufw allow 443`
        - Reload the firewall to apply the changes
          `sudo ufw reload`
        - Verify the firewall rules
          `sudo ufw status`
    - Verify if UFW is allowing traffic on ports 80 and 443:
      `sudo ufw status`

Database Service:

- **InfluxDB**: Database service is set so that other services can utilize it for storing data
    - Pre-configured with secure authentication.
    - Accessible to the backend for data storage and retrieval.
    - Verify by below steps:
        - Check that InfluxDB is running:
          `sudo systemctl status influxdb`
        - Ensure InfluxDB is listening on the correct ports (e.g., 8086) : `sudo ss –tuln | grep ':8086'`
        - Test Writing Data to the Database
        - Verify the data is stored correctly by querying the database.

---

5. Backend Data Collection

Key Features: This project implements REST API using NodeJS runtime and ExpressJS library. Once API endpoint is created for data collecting, it will be set as a service in the cloud and the API is then published by using Nginx webserver's reverse proxy.

- **Data Reception**:
    - IoT sensors send periodic temperature readings to the backend API.

        - Logs are available for debugging and monitoring data flow.
    - **Data Sanitization**:
        - Incoming data is validated to ensure integrity and prevent security risks.
    - **Database Integration**:
        - Sanitized data is written into InfluxDB efficiently.
        - Each entry is timestamped and categorized for easy querying.
    - The API Endpoint: /api/v1/embed. This API endpoint listens to HTTP GET requests on the path /api/v1/embed. It expects a query parameter called value, which contains the value that we wish to store in the database.

Backend Logs

- Logs show detailed information about incoming requests, ensuring observability: for example: [INFO] Data received: {"temperature": 25.3} [INFO] Data written to InfluxDB.
- The API Endpoint: /api/v1/temp endpoint is a GET API designed to retrieve temperature data stored in the InfluxDB database. It queries data from a specific bucket, filters it for the desired measurement and field, and returns the results in a JSON format for easy use in our frontend visualizations.

---

6. User Interface for Data Observation

Features

- **Grafana Dashboard**: see at http://localhost:3000
- Displays real-time temperature readings in an intuitive interface.
- Historical trends are visualized with interactive charts, enabling easy analysis of temperature patterns.

Embeding the Grafana to UI using iFrame

How to Access

1. **Real-Time View**:

- Visit the Grafana dashboard at `http://20.238.12.185/` or `myiot.northeurope.cloudapp.azure.com` to see live temperature data updates.

2. **Historical Data**:

- Visit the Grafana dashboard at `http://20.238.12.185/` or `myiot.northeurope.cloudapp.azure.com` to see historical temperature data updates.

** Conclusion **

This IoT Temperature Monitoring System leverages cutting-edge cloud technologies and software tools to provide a scalable, reliable, and user-friendly solution for monitoring temperature data. By combining the power of Azure, Node.js, InfluxDB, and Grafana, this system ensures efficiency, security, and accessibility for users worldwide.

Try It Out Visit the system at `http://20.238.12.185/` or `myiot.northeurope.cloudapp.azure.com` to explore real-time temperature monitoring and trends!

For any issues or inquiries, please contact [phu.trieu@student.lab.fi].

Clone the Repository**

```
git clone https://github.com/Phu-trieu24/Final-Task---IoT-Pipeline---Smart-Temperature-Sys
```

Or visit Project's Repository:

(https://github.com/Phu-trieu24/Final-Task---IoT-Pipeline---Smart-Temperature-System)

YouTube video - Introduce My IoT system ** See here:

(https://youtu.be/nOkosQe_zGo)

Author and Ownership: Phu Trieu