IOT MONITORING PLATFORM

Duțu Alin Călin

CONTENTS

1 INTRODUCTION

1.1 Context

The Internet of Things (IoT) is an engineering concept that refers to the network of physical objects embedded with sensors, software, and other technologies that are capable of connecting and exchanging data with other devices and systems over the Internet. These "smart" objects can collect, send, and act without human intervention, based on the acquired data. To analyze the data, it is required to create a pipeline that gathers the sensor data and processes it to provide a useful output.

1.2 Objectives

This program aims to create a simple pipeline that receives the data from sensors, stores and processes it to provide an output in an interactive graph form where users can visualize and analyze data distribution and draw conclusions or further process the data for diverse purposes.

The following objectives are covered upon implementing the pipeline:

- Understand the mechanics of the MQTT protocol, a well-known protocol in the IOT industry
- Understand the concepts of virtualization and interconnectivity between virtual machines
- Explore numerous tools and their impact on a multi-functional pipeline

2 PROPOSED SOLUTION

This project aims to implement a pipeline of apps for collecting, storing, and visualizing data received from a significant number of IOT devices. The architecture for the app pipeline implies a simplified version of public cloud services used in real-world applications. However, it is still designed with efficiency as a main objective for this project.

Looking from a general point of view, there are 4 services that perform different actions and interactions to ensure data processing:

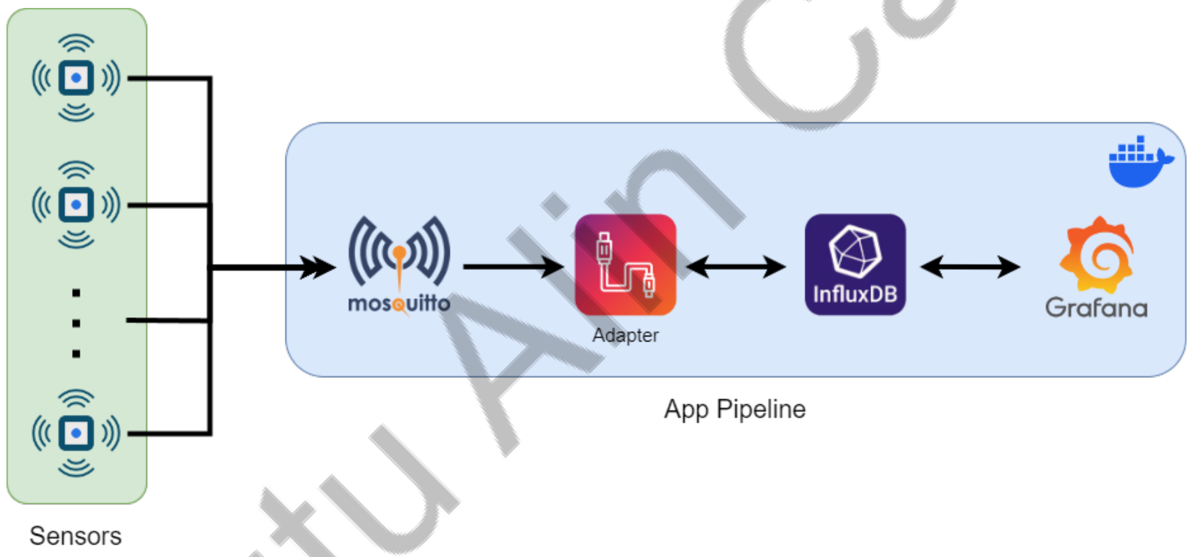

Figure 1: Application architecture [1]

The sensors are represented by programs provided by the SPRC team [1] that simulate the behavior of a real sensor and send pre-defined data as input to the app pipeline. The sensors can be used as a reliable testing input to validate the pipeline's output.

The App pipeline consists of 4 services:

- Mosquitto broker - an open-source tool that manages the sensor subscription and data transmission through topics.[2].
- Adapter - Gathers all the data received by the broker and sends it to the InfluxDB database.

[1]`https://www.flaticon.com/icons`. Icons created by RawPixel

- InfluxDB - Stores the data transmitted by the adapter making it available to the Grafana tool through the provided API[3].
- Grafana - Accesses data stored in the InfluxDB database to create charts for visualizing data distribution over time given the set configuration[4].

The communication between these 4 services is covered by Docker[5], a well-known platform designed to help developers build, share, and run application instances in containers. The enumerated services are built into isolated containers connected through internal networks that ensure data transmission between services. The app is released in clusters, one for each service, using the Docker Swarm mode[6].

3 IMPLEMENTATION DETAILS

The proposed solution consists of an application pipeline that gathers data provided by multiple IoT sensors, stores it in a local database configured specifically for various IoT data streams, and creates a dashboard with charts for visualizing data distribution over time.

3.1 Message broker

The message broker is implemented using the Mosquitto broker, an open-source tool for message distribution that implements the MQTT protocol's[7] broker.

After initialization, the broker is ready to receive 2 types of requests. The first one is from sensors that transmit data to topics. the broker is going to create a new topic if it hasn't already been created and associate the sensor data with it.

The second type of request is from the adapter which will ask to subscribe to all the topics created by the sensors. As soon as the broker executes the request, the adapter is going to receive the stored data from all the topics and all the upcoming updates.

3.2 Adapter

The adapter is a program written in Python used to adapt the sensor data stream to a format accepted by the InfluxDB database.

In the initialization process, the adapter sets a connection with the database first and then sends a request to the broker to subscribe to all topics. After initialization, the adapter waits for the broker to send messages with sensor data. As soon as one is received, it will be converted to the following JSON format and then sent to the InfluxDB database:

```
{
    "measurement": statie.cheie (string),
    "tags": {
        "location": locatie (string),
        "station": statie (string)
    },
    "time": timestamp (string),
    "fields": {"value": valoare (float)},
}
```

3.2.1 Database

The database used is an instance of InfluxDB, a vertical scaling database capable of withstanding multiple data streams that send small-sized sensor data in a short amount of time, such as multiple IoT devices[3]. The database stores data as the adapter calls the InfluxDB API to store the processed sensor data that is going to be used by Grafana to create the charts for data visualization.

3.2.2 Grafana

For data visualization, Grafana[4] is a great choice as it is an open-source analytics and monitoring solution capable of connecting and gathering sensor data from the InfluxDB database and creating dashboard and configurable charts, either programmatic through JSON configurations or manually on the user interface, to visualize data over a certain amount of time.

3.2.3 Docker

Docker[5] is the main component that makes the pipeline creation possible. Docker is a platform designed to build, share, and run applications in containers. In this case, Docker helps to build apps and deploy them as services in isolated containers, create internal network connections between them as shown in the architecture diagram [Fig. 1], expose the broker service to the outside world, and monitor the overall pipeline activity. Along with the Docker Swarm mode [6], the app can be easily monitored and released to production.

3.3 Testing

For testing the pipeline's reliability and software functionalities, the following methods have been used:

- Smoke tests - Testing that all pipeline components are operational and connected
- Stress tests - Testing the computational power to the limit by using multiple instances of the same sensor program
- Fuzz testing - Testing the modules with different inputs to check for any inconsistencies

4 CONCLUSIONS

In conclusion, this program aims to create a simple pipeline designed to gather data from multiple IoT sensors and process it to provide an output in chart forms where the user can visualize the data distribution over a certain period of time, and draw conclusions or further process the data for other purposes.

Bibliography

[1] SPRC team @ Polytechnic University of Bucharest. Systems for Programming Network Computers course. `https://curs.upb.ro/2022/mod/resource/view.php?id=45681`. The course is written in Romanian; Latest access: 14.07.2024;.

[2] Eclipse Foundation. Eclipse Mosquitto. `https://mosquitto.org/`. Latest access: 14.07.2024.

[3] InfluxData. InfluxDB. `https://www.influxdata.com/`. Latest access: 14.07.2024.

[4] GrafanaLabs. Grafana. `https://grafana.com/`. Latest access: 14.07.2024.

[5] Docker. Docker. `https://www.docker.com/`. Latest access: 14.07.2024.

[6] Docker. Docker Swarm Docs. `https://docs.docker.com/engine/swarm/stack-deploy/`. Latest access: 14.07.2024.

[7] Eclipse Foundation. MQTT prtocol. `https://mqtt.org/`. Latest access: 14.07.2024.