Anomaly Detection And Alerts: Node-RED, Mosquitto, and Docker

Refer to the IoT Module (Course 3) Week 3 practice exercise involving anomaly detection and alerts using Node-RED and Mosquitto.

That exercise builds upon the following prerequisites, covered in the related mentoring content:

1. Introduction to Node-RED
2. Creating MQTT Pub-Sub nodes on Node-RED
3. Running the script and publishing the data over the MQTT Broker
4. Anomaly detection
5. Data storage on MongoDB

We have modified the core tasks of the original Practice Exercise, to use **Docker** containers to run the Node-RED server and MQTT broker, in the AWS cloud.

- Run the MQTT broker as a **Docker image**. It is readily available in public repositories on the web, such as Docker Hub: https://hub.docker.com/_/eclipse-mosquitto
    - You can run this Docker container on an EC2 Instance.
    - You can initially store the MQTT Docker Image in AWS ECR, and then install it on the above EC2 instance.
    - You need to run with a device_id as the topic.
    - Push data from each of these topics individually, to a **DynamoDB** database.
    - Instead of using the actual devices, you can try out the exercise with Python simulation code as well.
- Run Node-RED server as a **Docker image**. It is readily available in public repositories on the web, such as Docker Hub: https://hub.docker.com/r/nodered/node-red-docker/
    - You can run this Docker container on a separate EC2 instance.
    - You can initially store the Node-RED Docker Image in AWS ECR, and then install it on the above EC2 instance.
    - Read in the pushed device data from the **DynamoDB** database.
    -

- Visualize the pushed data based on their device_id's by creating a User Interface over Node-RED.
- Configure the User Interface to perform deeper analysis on the collected data.
- Based on the anomaly data received, check if we can create an alert mechanism to inform the end-user.