



Smart Cities Hackathon Tier 2 Challenge 6

Challenge 6: Configure an NB-IoT Sensor, or Virtual NB-IoT Sensor

In this challenge, you can either connect one of the supplied NB-IoT Sensors from AllIoT or create a docker image and Node script to connect your virtual sensor from Challenge 1 to mqtt.core.aql.com. Both approaches follow the same principle of a sensor connecting over MQTT and securely uploading JSON readings for charting. If you don't know what MQTT is ask the AI and it will provide details.

To complete this challenge you will need your Location ID and Device ID which can be sourced from <https://core.aql.com/login>

Step 1

If you would like to create a virtual NB-IOT Sensor then skip to step 2

If you are using the supplied NB-IoT Sensor, you will need to create a new **Other** Decoder and Device first in core.aql.com and update the mqtt details using the required APP, The NB-IoT Sensor is an Adeunis sensor (<https://www.adeunis.com/en/>) they upload a JSON object

The MQTT Username and password will be supplied on the day, the host and port details are below

Host: mqtt.core.aql.com
Port: 8884

The CA public certificate can be downloaded from <https://trusti.uk/certificates>, you will require the intermediate certificate.

Step 2

Go to the official Docker website: <https://www.docker.com/products/docker-desktop>
Download Docker Desktop for your operating system (Windows, macOS, or Linux).
Follow the installation instructions provided for your operating system.

Verify Docker installation:

After installing Docker Desktop, open a terminal or command prompt.

Install Docker desktop onto your machine.

Step 2

Create a new folder for your Tier 2 challenges.

Step 3

Create a **dockerfile**

```
# Use the official Node.js image as a base
FROM node:latest
```

```
# Set the working directory in the container
WORKDIR /app
```

```
# Copy package.json and package-lock.json (if available)
COPY package*.json ./
```

```
# Install dependencies
RUN npm install
```

```
# Copy the rest of the application code
COPY . .
```

```
# Command to run the Node.js script
CMD ["node", "hackathon.js"]
```

Step 4

Create a **compose.yml** this provides the name for our container, feel free to change the name

```
services:
  node:
    build: .
    container_name: hackathon
    restart: always
```

Step 5

To run the scripts over MQTT we require a package for this so we next need to define the **package.json**

```
{
  "name": "hackathon",
  "version": "1.0.0",
  "description": "",
  "main": "hackathon.js",
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "mqtt": "^4.2.6"
  }
}
```

Step 6

We next need to create a new **hackathon.js** file , the first part of the script defines the constants and MQTT details. You will need to replace your location id, device id and team name. The Username and Password details are available on the day

```
const mqtt = require('mqtt');
const LOCATION_ID = "YOUR_LOCATION_ID";
const DEVICE_ID = "YOUR_DEVICE_ID";
const TOPIC = `location/${LOCATION_ID}/device/${DEVICE_ID}/event/up`;
const TEAM = "YOUR_TEAM"
const USER_NAME = "TO BE REPLACED"
const PASSWORD = "TO BE REPLACED"
```

```
let mqttConnectionDetails = {
  host: "mqtt.core.aql.com",
  port: 8884,
  protocol: "mqtts",
  keepalive: 10,
  clientId: TEAM,
  protocolId: "MQTT",
```

Tier 2: Challenge 6



```
protocolVersion:    4,  
clean:              true,  
reconnectPeriod:    2000,  
connectTimeout:     2000,  
rejectUnauthorized: false,  
username: USER_NAME,  
password: PASSWORD  
}
```

Step 7

Next we need to create a function to generate a virtual payload as per your virtual sensor in challenge 5

```
function generatePayload() {  
  // Generate random values for each field  
  const timestamp = new Date().toISOString();  
  const latitude = Math.random() * (90 - (-90)) + (-90); // Random latitude between  
-90 and 90  
  const longitude = Math.random() * (180 - (-180)) + (-180); // Random longitude  
between -180 and 180  
  const temperature = Math.random() * (40 - (-20)) + (-20); // Random temperature  
between -20 and 40  
  const humidity = Math.random() * (100 - 0) + 0; // Random humidity between 0  
and 100  
  const co2 = Math.random() * (1000 - 400) + 400; // Random CO2 between 400  
and 1000  
  const pm25 = Math.random() * (100 - 0) + 0; // Random PM2.5 between 0 and  
100  
  const pm10 = Math.random() * (100 - 0) + 0; // Random PM10 between 0 and 100  
  
  // Return the payload object  
  return {  
    timestamp: timestamp,  
    latitude: latitude,  
    longitude: longitude,  
    temperature: temperature,  
    humidity: humidity,  
    co2: co2,  
    pm25: pm25,  
    pm10: pm10  
  };  
}
```

```
}
```

Step 8

Now we have the readings we need to connect to the MQTT broker and upload a reading every minute

```
// Connect to MQTT broker with authentication  
const client = mqtt.connect(mqttConnectionDetails);  
  
// When connected, publish the payload to the specified topic every minute  
client.on('connect', () => {  
  console.log('Connected to MQTT broker');  
  
  // Initial upload upon connection  
  const initialPayload = generatePayload();  
  console.log("Initial Payload:", initialPayload);  
  console.log('Uploading to topic:', TOPIC);  
  client.publish(TOPIC, JSON.stringify(initialPayload), () => {  
    console.log('Initial message published');  
  });  
  
  // Upload every minute  
  setInterval(() => {  
    const payload = generatePayload();  
    console.log("Payload:", payload);  
    console.log('Uploading to topic:', TOPIC);  
    client.publish(TOPIC, JSON.stringify(payload), () => {  
      console.log('Message published');  
    });  
  }, 60000); // 60000 milliseconds = 1 minute  
});  
  
// Log errors  
client.on('error', (error) => {  
  console.error('MQTT error:', error);  
});
```

Step 9

Next, build your docker file, this may take a few minutes to pull the base image

docker compose up --build

Step 10

Log into core.aql.com and confirm you can see the readings being displayed. You have now completed this challenge