**Complete AWS configuration guide in building the Thermostat Dashboard**

**Sensor Watch**

**Overview:** The SensorWatch Dashboard is a static web application designed to visualize temperature and light sensor data collected by the RNWF IoT device. This dashboard relies on various AWS Cloud services for data collection, storage, and retrieval, enabling users to monitor real-time sensor data seamlessly.

#### **Key Features:**

* **Data Visualization**: Real-time plotting of temperature and light sensor data.
* **User Authentication**: Secure user access and data management.

#### **AWS Cloud Services Utilized**

1. **AWS IoT Core**:
   * **Device Management**: Each connected IoT device is referred to as an "IoT Thing" and has a unique "Thing name" and device certificate.
   * **Message Publishing**: IoT devices publish data to the cloud using MQTT publish topics, formatted as iot/Thingname/sensor, where "Thingname" varies per device.
   * **IoT Rules Engine**: This engine selects specific messages based on criteria and performs actions such as storing data in DynamoDB.
2. **AWS DynamoDB**:
   * **Data Storage**: DynamoDB tables store all records (items) received from AWS IoT Core.
   * **Partition Key**: Each item is uniquely identified by a Partition Key, facilitating efficient data retrieval.
   * **Service Integration**: Other AWS services, such as AWS Lambda, can query DynamoDB tables using the Partition Key to access specific records.
3. **AWS Lambda**:
   * **Functionality**: Lambda functions interact with other AWS services to perform specific tasks.
   * **User Interaction**: In this application, a Lambda function is triggered by user actions from the front-end dashboard, enabling dynamic data handling and processing.
4. **Amazon Cognito**:
   * **User Authentication**: Provides secure user authentication, authorization, and user management, ensuring only authorized/specific users can access the dashboard. In our application guest access is granted without authorization for multiple users.
5. **Amazon S3**:
   * **Web Hosting**: Hosts the static web application, ensuring high availability and scalability for the SensorWatch Dashboard. The Dashboard files are upload into the bucket and dynamically accessed using S3 bucket URL.

#### **Workflow**

1. **Data Collection**: The RNWF11 WiFi click on the IoT device establishes an MQTT connection with AWS IoT Core using MQTT-based AT commands.
2. **Data Publishing**: IoT devices publish sensor data to a specified MQTT topic, which includes the Thing name.
3. **Rule Processing**: The IoT Rules engine processes incoming messages and stores relevant data as items in a DynamoDB table.
4. **Data Storage**: DynamoDB stores sensor data records, making them accessible to other services.
5. **User Interaction**: Users interact with the dashboard, triggering AWS Lambda functions that query DynamoDB to fetch and display specific sensor data.

A diagram of a cloud

Description automatically generated

**AWS Account setup**

**(1) Create an Admin AWS Account**

An AWS account, preferably, with administrator access privileges is required for enabling the user to leverage the AWS Cloud services for demonstrating the Wireless Thermostat application with RNWF11.

1.If the user does not have an AWS account, they can create a root account by following the steps mentioned [here](https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/).

It is recommended not to use the root user for everyday or administrative tasks. Instead, adhere to the best practice of using the root user only to create the first IAM user. The user must **create an admin IAM user account for themselves** to protect the root user account and use them to perform only a few account and service management tasks.

2. The steps to create an admin IAM user can be on Amazon’s “Step 1: [Set Up an AWS Account and Create an Administrator User](https://docs.aws.amazon.com/streams/latest/dev/setting-up.html)” page.

Note: The username and password can be chosen by the user.

3. The user can log into their admin IAM account using the IAM credentials provided in the previous step. Select a region from the region drop down list.(E.g. Us-east-2(Ohio))

**(2) AWS Lambda Setup**

**For Creating Lambda function to fetch IoT Data**

1. Go to the AWS Lambda service. Click on **the Create function**.

A white rectangle with a black line

Description automatically generated

2. Select **Author from scratch > Function name >sensor\_data\_fetch.**

**Note:** User can select any Function name of their choice.

3. Select **Runtime > Python 3.8 or higher**.

4. Select the execution role as **Create a new role with basic Lambda permissions** and select “**Create function**”.

Lambda will create an execution role named sensor\_data\_fetch-role-, with permission to upload logs to Amazon CloudWatch Logs. The function creation will take some time.

5. The following page appears when the function is created.

A screenshot of a computer

Description automatically generated

6. In the Function dashboard, navigate to the **Configuration** tab and click on the **Execution** role that was created ,under **Permissions**.

A screenshot of a computer

Description automatically generated

7. You will be taken to the IAM Roles page of the above execution role. Click on **Add Permissions >Attach Policies**.

A screenshot of a computer

Description automatically generated

8. Search for “AmazonDynamoDBFullAccess” in the search box and choose “Add permissions”.

9. Now you can see the list of permissions for the sensor\_data\_fetch-role as below.

A screenshot of a computer

Description automatically generated

10. Go to the Lambda function’s Dashboard and copy the following lambda handler function inside the ‘code source’ section under “**Code**” tab.(Please refer the image below)

*import boto3*

*def lambda\_handler(event, context):*

*# Specify the AWS region*

*region\_name = 'us-east-2' # Replace 'us-east-2' with the actual AWS region name*

*# Initialize the DynamoDB client*

*dynamodb = boto3.resource('dynamodb', region\_name='us-east-2')*

*# Get a reference to the DynamoDB table*

*table\_name = 'sensordata\_mchpmasters' # Replace 'YourTableName' with the actual name of your DynamoDB table*

*table = dynamodb.Table(table\_name)*

*try:*

*# Extract the device ID from the event payload*

*device\_id = (event.get('DeviceID')) # Assuming 'DeviceID' is the key in the event payload*

*print(device\_id)*

*if 1: #<= device\_id <= 10:*

*# Retrieve the item from DynamoDB based on the device ID*

*response = table.get\_item(*

*Key={*

*'Thing\_name': device\_id*

*}*

*)*

*payload = response.get('Item', {})*

*print("Obtained Item:", payload)*

*temperature = payload.get('payload', {}).get('temperature')*

*timestamp = payload.get('payload',{}).get('Timestamps')*

*light=payload.get('payload',{}).get('light')*

*print("tempertaure:", temperature)*

*print("timestamp:",timestamp)*

*if temperature is not None:*

*# Return the temperature value*

*return {*

*'statusCode': 200,*

*'body': {*

*'temperature': temperature,*

*'timestamp':timestamp,*

*'light':light*

*}*

*}*

*else:*

*# Temperature attribute not found in the item*

*return {*

*'statusCode': 404,*

*'body': {*

*'message': f"No temperature data found for DeviceID: {device\_id}"*

*}*

*}*

*except Exception as e:*

*# Handle any errors*

*print(f"Error retrieving temperature data from DynamoDB: {str(e)}")*

*return {*

*'statusCode': 500,*

*'body': {*

*'message': f"Error retrieving temperature data from DynamoDB: {str(e)}"*

*}*

*}*

A screenshot of a computer

Description automatically generated

1. The sensor\_data\_fetch lambda function will be updated after clicking “**Deploy**” button.

A screenshot of a computer

Description automatically generated

1. Now, the Lambda function is configured to retrieve the IoT data for multiple IoT devices from the DynamoDB table.

### **(3)Amazon Cognito Setup**

**Create an Amazon Cognito user pool**

1. Open Amazon Cognito console and choose **“Create a Userpool”**.
2. A screenshot of a computer screen

   Description automatically generatedChoose “**Username**” from the cognito sign-in attribute options and click “**Next**”.
3. Choose “**No MFA**” and click “**Next**”. Leave other settings in their default settings.
4. In the “**Configure sign-up experience**” , click “**Next**” with default settings.
5. In the “**Configure message delivery**” page, choose “**Send e-mail with Cognito**” and click “**Next**”.

A screenshot of a computer

Description automatically generated

1. Enter a user pool name of your choice such as “**sensordata\_mchpcognitouserpool**”.A screenshot of a computer

   Description automatically generated
2. Enter an AppClient name of your choice such “**MasterClient**”. Review and create the userpool.

A screenshot of a computer

Description automatically generated

**P.S.** Please make note of the Userpool id for adding it in the identity pool.

**Create an Amazon Cognito identity pool**

1. Open the Amazon Cognito console and choose **Identity Pools**.
2. Create a new identity pool.
3. Choose “**Authenticated Access**” and “**Amazon cognito userpool”** as the authenticated identity source and click “**Next**”.

A screenshot of a computer

Description automatically generated

1. Choose “**Create a new IAM role**” and enter an IAM role name of your choice.(such as sensordata\_mchpmasterscognitoguestrole).Click “**Next**”.

A screenshot of a computer screen

Description automatically generated

1. Enter the userpool id and appclient id created earlier. Click “**Next**”.(The appclient id can be found in the “App Integration” tab of the userpool previously created.)
2. Enter an Identity pool name such as “**sensordata\_identitypool** “and click “**Next**”. Review and create the identity pool.

**Create and attach an IAM policy to the cognito identity**

1. Open the IAM console, and from the navigation pane, choose **Roles**.
2. Choose the Guest Role previously created (sensordata\_mchpmasterscognitoguestrole).
3. A screenshot of a computer

   Description automatically generatedAdd **AWSLambdaFullAccess** and **AmazonS3FullAccess** permissions to this role.

**(4) Setup Amazon DynamoDB**

**Create a table:**

1. Go to the Amazon DynamoDB service.

2. Choose **Create Table** from **Tables**.

3. In the **Create DynamoDB Tables** screen, do the following:



4. In the **Table** name box, enter a table name such “**sensordata\_mchpmasters**”. You may enter a table name of your choice. However**, be sure to use the same name in the table name of the sensor\_data\_fetch Lambda function**.

5. For the Primary key, in the **Partition key** box, enter “**Thing\_name**”. Set the data type to **String**.

6. Leave the Sort key unfilled.

7. When the settings are as you want them, choose **Create**.

**(5)Setup AWS IoT Core**

1. Go to AWS IoT Core services.

2. Under **Manage > All devices > Things**, click **Create Things**.

3. Select **Create a Single Thing**.

4. Enter the thing name of your choice and click **Next** (e.g., RNWF\_149).

5. Click on **Skip creating a certificate currently** and select **Create Thing**.

6. Under **Secure > Certificates**, click on **Add Certificate > Register Certificates**.

7. Select **CA is not registered with AWS IoT**.

8. Upload the device certificate (.crt).

9. Click **Activate all** and click on **Register Certificate**.

10. Under **Security > Policies**, select **Create a policy**.

11. Enter a policy name of your choice (e.g.,RNWF11\_policy).

12. Copy the following resource ARN policies for IoT Connect, Publish, Subscribe and Receive policies.

**Connect ARN**: \*

**Publish ARN**: \*

**Receive ARN**: \*

**Subscribe ARN**: \*



13. Select **Allow** under **Effect** and click **Create**.

14. Go to **Security > Certificates** and select the certificate we created.

15. Under **Policies** tab, select **Attach Policy** and select the policy created.

16. Under **Things** tab, select **Attach to Things** and select the thing name we created for the certificate to get attached to the thing.

**Creating AWS IoT Rule**

1. In the AWS IoT Navigation pane, go to **Message Routing > Rules**. Select **Create Rule**.

2. Enter Rule name “sensordata\_dynamodbrule”. You can use any rule name of your choice. Click “Next”.

3. The Rule must be set using SQL query as: **SELECT temperature,light,timestamp() AS Timestamps FROM 'mchp-iot3/deviceupdate/RNWF\_XXX'**

Click Next.

4. Under **Rule Action** select **DynamoDB** as **Action 1** and choose the from the DynamoDB table dropdown list.

5. In **Partition key**, enter “**Thing\_name**”.

6. In **Partition key value**, enter **“${cast(topic(3) AS VARCHAR)}“.**This is the first substitution template you will use in this rule. Instead of using a value from the message payload, it will use the value returned from the topic function.

7. The default Operation is INSERT. Other fields can be kept blank. In **IAM role**, choose **Create new role**.

8. You can create a new role with **full access permission to DynamoDB**.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

9. Review the entered details and create the rule.

**Setup AWS S3 Bucket**

1. Go to AWS S3 services.
2. Select “**Create Bucket**”.

A screenshot of a computer

Description automatically generated

1. Enter a bucket name of your choice such as “**Thermostatbuck**”.
2. Select “**ACLs enabled**”.

A screenshot of a computer

Description automatically generated

1. **Uncheck “Block all public Access**”.

A screenshot of a bucket

Description automatically generated

1. **Create Bucket**.

**Setting Bucket Permissions**

1. Go to “Permissions” tab inside the created bucket.

A screenshot of a computer

Description automatically generated

1. Click on edit the Bucket policy and copy the following JSON bucket policy and save it.

*{*

*"Version": "2012-10-17",*

*"Statement": [*

*{*

*"Sid": "PublicRead",*

*"Effect": "Allow",*

*"Principal": "\*",*

*"Action": [*

*"s3:GetObject",*

*"s3:GetObjectVersion",*

*"s3:PutObject"*

*],*

*"Resource": "arn:aws:s3:::**Thermostatbuck /\*"//Replace with your bucket name*

*}*

*]*

*}*

1. Go to CORS and edit the configuration as below.

*[*

*{*

*"AllowedHeaders": [*

*"\*"*

*],*

*"AllowedMethods": [*

*"GET",*

*"PUT"*

*],*

*"AllowedOrigins": [*

*"\*"*

*],*

*"ExposeHeaders": [],*

*"MaxAgeSeconds": 3000*

*}*

*]*

1. Edit the ACL(Access control list) to enable **List and Read access.**

A screenshot of a computer

Description automatically generated

A white card with black text

Description automatically generated

1. Save the changes.

**Message to be sent from the IoT device.**

1. The RNWF11 IoT device should send the message in the following **JSON format**.

**{**

**“temperature”: <temp value>,**

**“light”:<light value>**

**}**

Here, <temp value> and <light value> are the actual temperature and light sensor values published to the AWS IoT cloud.

1. The message should be published to **publish topic** “mchp-iot3/deviceupdate/RNWF\_XXX”. Please note that the ‘+’ in the publish topic should be replaced with the Thing name of that specific device to support multiple device IoT infrastructure.

For example, if the Thing name is RNWF\_149, the publish topic should be “mchp-iot3/update/RNWF\_XXX” and so on.

**P.S.(1)** For initial testing the AWS MQTT client can be used for publishing the messages of the above format.

1. The timestamp is dynamically auto computed by the AWS Rules engine and is not present in the actual JSON message published to the IoT device.

**S3 Bucket File Upload**

1. **Viewing Dashboard**

The S3 bucket created in the previous steps is used to store the Dashboard source files that includes .html,.js and .jpg files. The S3 bucket objects are granted public access and hence the bucket object URL of the html file can be used for viewing the temperature and Light sensor Dashboard.

Object URL : <https://xxxbucket.s3.us-east-2.amazonaws.com/SensorWatch.html> //Changes with different bucket

**This object URL is shared to users for visualizing the Dashboard. It is granted public access.**

The Dashboard source files include

1. SensorWatch.html
2. Thermostat\_helper.js
3. MASTERs2024\_logo1.jpg
4. MASTERs2024\_logo2.jpg
5. Utilis.js

All these files should be uploaded by user into the S3 bucket using the “**Upload**” option.

A screenshot of a computer

Description automatically generated