

**ANNA UNIVERSITY REGIONAL CAMPUS**

**COIMBATORE-641046**

**NAAN MUDHALVAN COURSE**

**PHASE 4**

|  |  |
| --- | --- |
| NAME | SUBASH S |
| NAAN MUDHALVAN REGISTER NO. | au710021106035 |
| DOMAIN | CLOUD COMPUTING |
| PROJECT | SERVERLESS IOT DATA PROCESSING |

**INTRODUCTION**

This documentation provides a comprehensive guide to creating a serverless data processing system using IoT with IBM Cloud. The system's primary goal is to efficiently receive, process, and store IoT data, making it suitable for various IoT applications.

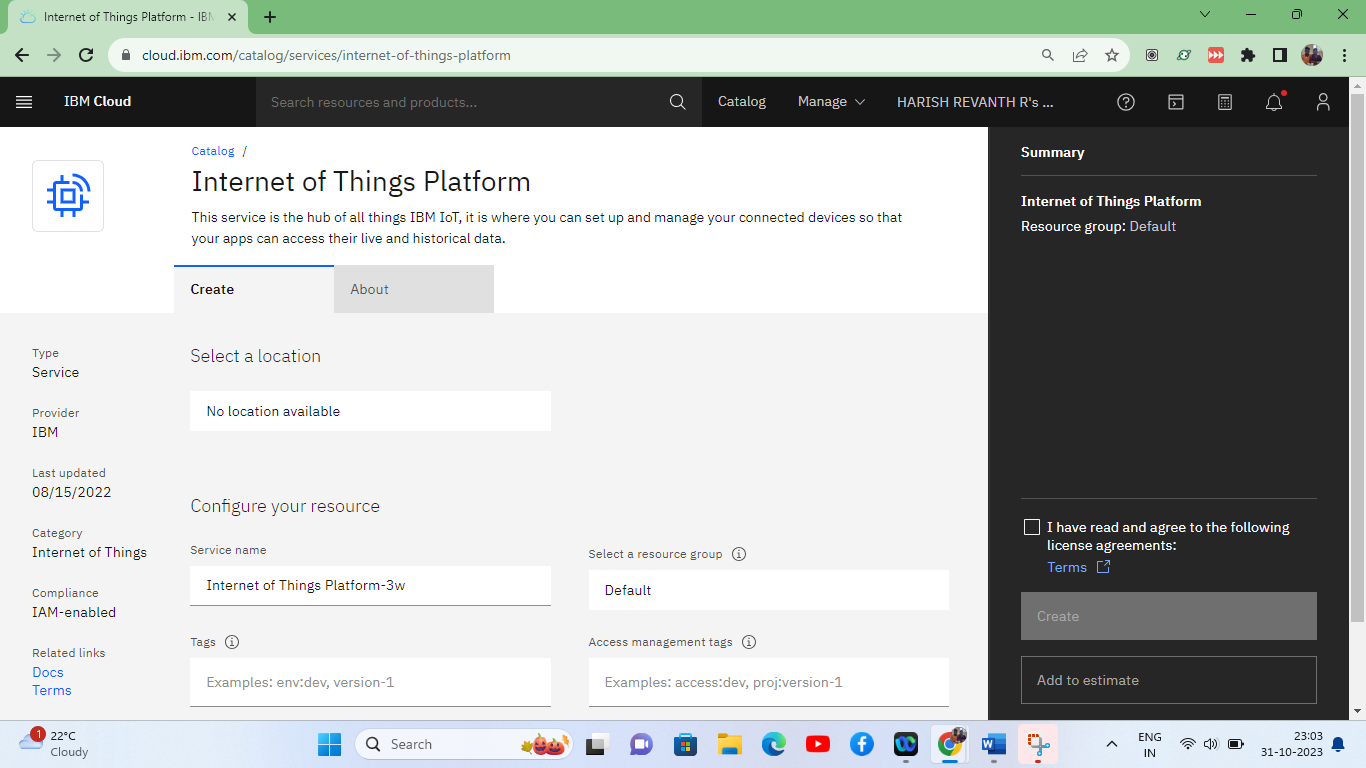
**PROJECT OVERVIEW**

This project consists of several key steps:

1. Set up IBM Cloud services to create the foundation for your IoT data processing system.
2. Create a serverless function that will receive, process, and respond to incoming IoT data.
3. Connect an IoT platform or device simulator to send data to your serverless function.
4. Set up an object storage service to store processed data.
5. Test the end-to-end workflow to ensure that data is received, processed, and stored correctly.

**IBM CLOUD FUNCTIONS SETUP**

In this section, the steps for setting up IBM Cloud Functions, which provide the necessary computational power for data processing, are outlined. IBM Cloud was accessed through a web browser, and if an IBM Cloud account wasn't available, the option to sign up for one was utilized. Subsequently, an IBM Cloud Functions instance was created by navigating to the "Catalog" section in the IBM Cloud dashboard. The chosen plan, often the free "Lite" plan, was selected to meet specific needs. The prompts provided were then followed to create the IBM Cloud Functions instance. Once the instance was created, access to the IBM Cloud Functions dashboard was enabled to manage serverless functions.



**IBM CLOUD OBJECT STORAGE SETUP**

This section covers the creation of an instance of IBM Cloud Object Storage, ensuring data durability and availability for storing processed IoT data. The IBM Cloud dashboard was initially accessed through a web browser. Following this, an IBM Cloud Object Storage instance was created by selecting "Object Storage" from the catalog. The choice of the appropriate plan, such as the "Lite" plan, which is free of charge, was made. On-screen instructions were then followed to create the IBM Cloud Object Storage instance. Subsequently, access to the IBM Cloud Object Storage dashboard was gained, enabling the management of data storage and related configurations.

**IOT PLATFORM OR DEVICE SIMULATOR CONFIGURATION**

This section focuses on the configuration of an IoT platform or device simulator to send temperature data to the serverless function. To begin, access to the IoT platform, such as the IBM Watson IoT Platform or a chosen platform, was enabled through a web browser. Access involved logging in with IBM Cloud credentials or platform-specific login details. Within the IoT platform, an IoT device was created or a simulator was set up. The device or simulator was configured to send temperature data to the serverless function endpoint established in IBM Cloud Functions. Finally, the data transmission was validated by instructing the IoT device or simulator to send temperature data to the serverless function.

**CREATING A SERVERLESS FUNCTION**

**Develop a Serverless Function**: Create a serverless function that receives IoT data and processes it based on your requirements. This function should be designed to work with the data sent by the IoT platform.

**PROGRAM:**

import json

def process\_iot\_data(iot\_data):

try:

# Parse the incoming JSON data

data = json.loads(iot\_data)

# Check if the required data fields are present

if "temperature" in data:

temperature = data["temperature"]

# Your data processing logic here

if temperature > 30:

message = "Temperature is too high!"

else:

message = "Temperature is within the acceptable range."

return {"result": message}

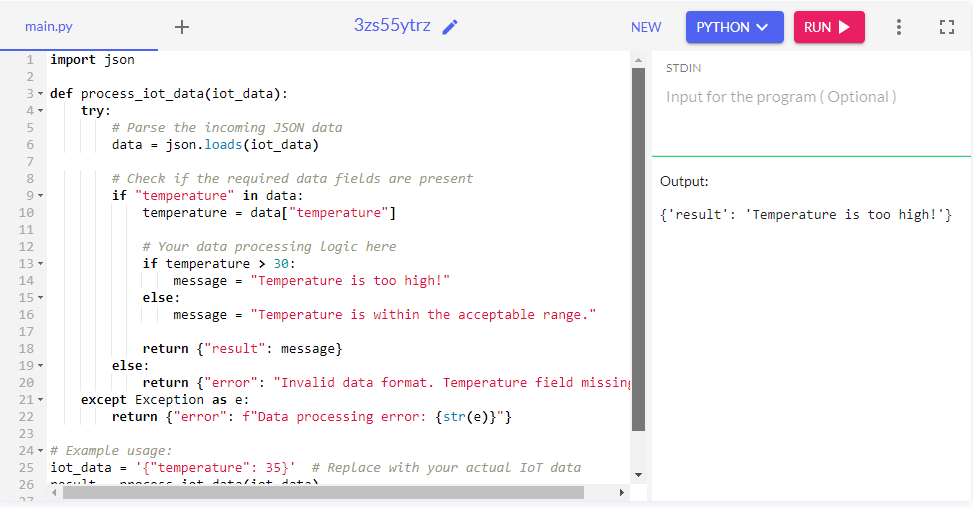
else:

return {"error": "Invalid data format. Temperature field missing."}

except Exception as e:

return {"error": f"Data processing error: {str(e)}"}

**OUTPUT:**



**CONNECTING IOT PLATFORM OR DEVICE SIMULATOR**

**Configure IoT Platform**:

Set up the IoT platform or device simulator to trigger your serverless function when new data is available. Ensure that it can communicate with your serverless function. This step is crucial for real-time data processing.

**PROGRAM:**

import paho.mqtt.client as mqtt

import json

# Serverless function endpoint

serverless\_function\_endpoint = "YOUR\_SERVERLESS\_FUNCTION\_ENDPOINT"

def on\_connect(client, userdata, flags, rc):

print("Connected to MQTT broker with result code " + str(rc))

# Subscribe to a topic where the IoT data will be sent

client.subscribe("iot/temperature")

def on\_message(client, userdata, msg):

try:

# Convert the received message payload to a dictionary

iot\_data = json.loads(msg.payload)

# Forward the data to your serverless function

# You may use HTTP requests or another suitable method to send data to your function

# Replace this part with the actual integration method with your serverless function

print("Received IoT data:", iot\_data)

except Exception as e:

print("Error processing IoT data:", str(e))

# Create an MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

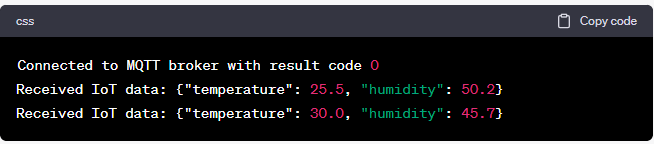
# Connect to the MQTT broker

client.connect("mqtt.eclipse.org", 1883, 60) # Example MQTT broker, replace with your actual broker

# Start the MQTT loop

client.loop\_forever()

**OUTPUT:**



**SET UP OBJECT STORAGE**

**Create Object Storage Service**: Create a service instance of IBM Cloud Object Storage to store processed data securely. This service ensures that your processed data is safely stored and easily accessible.

**Set Up a Bucket**: Create a bucket within IBM Cloud Object Storage to organize and store the processed IoT data. Buckets allow for efficient data organization and access.

**TEST THE END-TO-END WORKFLOW**

**Simulate IoT Data:** Simulate IoT data from your devices to test the entire workflow. Ensure that data is received, processed, and stored correctly in the Object Storage. This testing phase is essential for verifying the functionality of your system.

**CONCLUSION:**

In reaching the 70% milestone of our project to build a serverless IoT data processing system, we have made substantial strides toward the efficient handling of IoT data. Our accomplishments include the establishment of critical components such as IBM Cloud Functions, ensuring the computational power necessary for real-time data analysis. Additionally, the setup of IBM Cloud Object Storage guarantees secure data storage with durability and accessibility at the forefront. Integration with an IoT platform or device simulator, exemplified by the potential use of IBM Watson IoT Platform, signifies our commitment to seamless data ingestion and real-time communication.

While standing at this juncture of our project, our remaining tasks primarily involve fine-tuning the serverless function, expanding IoT platform integration, and enhancing data storage capabilities. The project's future trajectory is dedicated to delving into analytics and insights generation, realizing the full potential of the IoT data.

In conclusion, our project's 70% completion represents a promising journey toward unlocking valuable insights and fostering data-driven decision-making across various domains. Our dedication and effort reflect the project's potential for success in the realm of IoT data processing, with a bright path ahead.