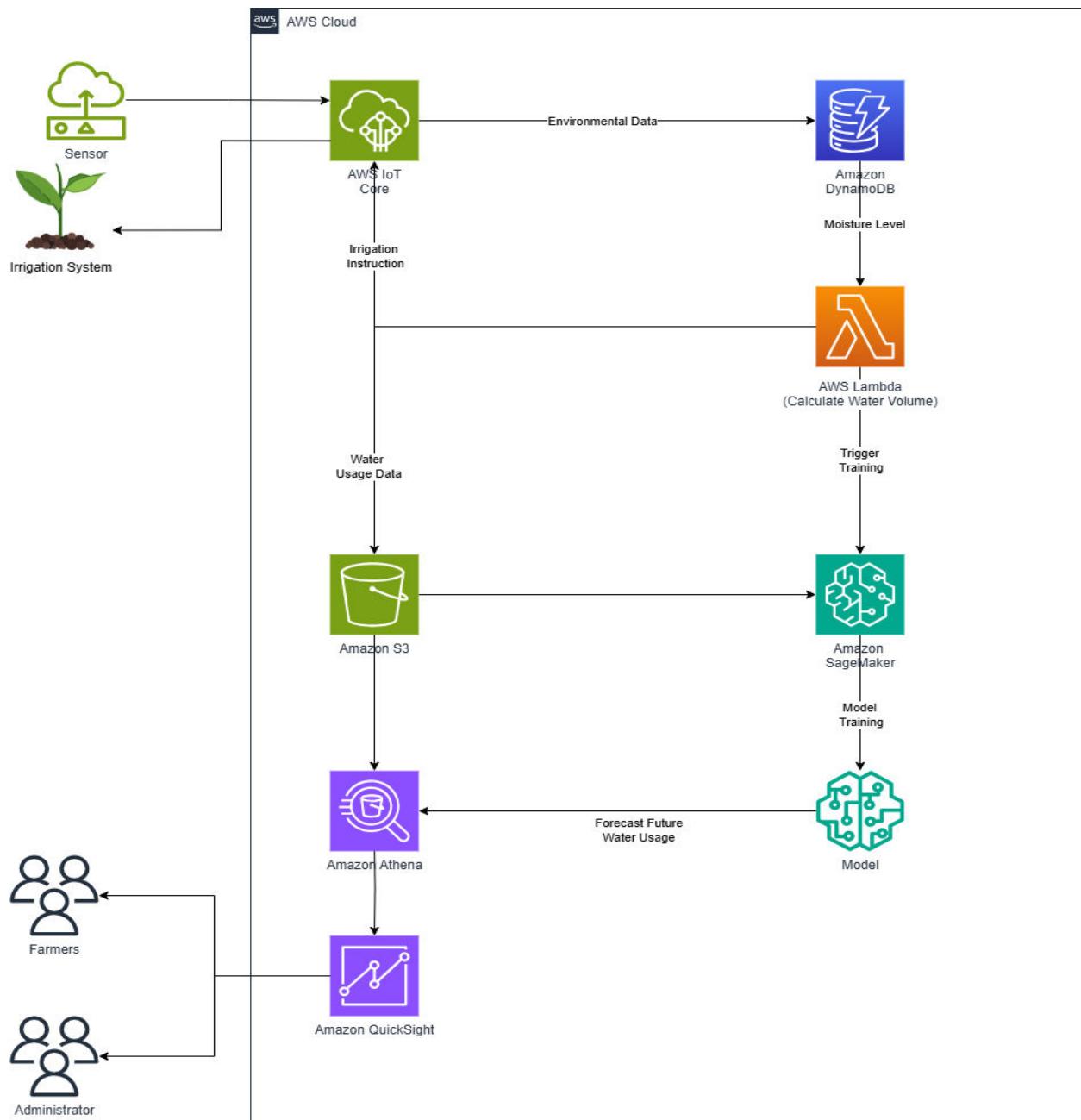


# Architecture Diagram



# AWS Management Console

# Sign in to the AWS Console

The screenshot shows the AWS Free Tier landing page. At the top right, there is a prominent red arrow pointing towards the "Sign In to the Console" button. The page features a dark header with the AWS logo and various navigation links like "About AWS", "Contact Us", "Support", "English", "My Account", and a search bar. Below the header, a purple bar contains links for "Amazon Q", "Products", "Solutions", "Pricing", "Documentation", "Learn", "Partner Network", "AWS Marketplace", "Customer Enablement", "Events", "Explore More", and a search icon. A secondary navigation bar below includes "AWS Free Tier", "Overview" (which is underlined), "Free Tier Categories", "How to Create an Account", "Featured Offers for Business", "FAQs", and "Terms and Conditions". A large, bold "AWS Free Tier" title is centered above a sub-headline "Gain free, hands-on experience with the AWS products and services". A "Create a Free Account" button is located on the left. On the right, there is a message bubble from an AWS representative and a yellow circular icon with a speech bubble containing the number "1".

Learn more about AWS Free Tier →

# AWS Free Tier

Gain free, hands-on experience with the AWS products and services

Create a Free Account

Hi, I can connect you with an AWS representative or answer questions you have on AWS.



Sign in as IAM user.

Try the new sign in UI

See our new improved Amazon Web Services sign in experience before we officially launch.

Enable new sign in

aws

### Sign in as IAM user

Account ID (12 digits) or account alias

IAM user name

Password

Remember this account

**Sign in**

[Sign in using root user email](#)

[Forgot password?](#)

aws

# Free, on-demand training

Boost your career with 600+ digital courses on AWS Skill Builder

[Learn more >](#)



# AWS Management Console

The screenshot shows the AWS Management Console Home page with the following sections:

- Recently visited:** A grid of recently used services including Amazon SageMaker, DynamoDB, IoT Core, IAM, Billing and Cost Management, Lambda, S3, AWS Health Dashboard, CloudWatch, EC2, and Simple Notification Service.
- Applications:** Shows 0 applications in the current region (ap-southeast-1). It includes a "Create application" button and a "Find applications" search bar.
- Welcome to AWS:** A section with a rocket icon and a link to "Getting started with AWS". It also includes a "Learn the fundamentals and find valuable information to get the most out of AWS." message.
- AWS Health:** Displays 0 open issues and 0 scheduled changes over the past 7 days.
- Cost and usage:** Shows current month costs at \$0.00 and forecasted month end costs at 0.

# AWS IoT Core

## Search IoT Core service from the search bar

The screenshot shows the AWS search interface with the query "IoT Core". A red arrow points to the "Services" link in the sidebar, which is highlighted with a black box. Another red arrow points to the "IoT Core" service card in the search results.

aws Services  X

☰ Services

Search results for 'IoT Core'

Services (42)      See all 42 results ►

Features (106)

Resources New

Documentation (122,104)

Knowledge Articles (461)

Marketplace (161)

Blogs (13,755)

Tutorials (36)

Events (297)

**Services**

**IoT Core** ☆  
Connect Devices to the Cloud

**AWS IoT Core for LoRaWAN**  
Connect, manage, and secure LoRaWAN devices at scale

**Amazon Fraud Detector** ☆  
Detect more online fraud faster using machine learning

**IoT Analytics** ☆  
Collect, preprocess, store, analyze and visualize data of IoT devices

**Features**

See all 106 results ►

In the left sidebar, click on 'Things' under 'All devices,' then click the 'Create thing' button to create a new IoT thing.

The screenshot shows the AWS IoT Things management interface. On the left, a sidebar lists various management options. Under the 'Manage' section, 'All devices' is expanded, showing 'Things' (circled with a red number 2). Below it are 'Thing groups', 'Thing types', and 'Fleet metrics'. Other collapsed sections include 'Greengrass devices', 'Software packages', 'Remote actions', 'Message routing', 'Retained messages', and 'Security'. The main content area displays a table titled 'Things (2) Info' with two entries: 'sensor\_region1a' and 'Sensor1'. A large orange 'Create things' button is located at the top right of the table. The top navigation bar includes the AWS logo, services menu, search bar, and account information for 'jianwei-aws' in Singapore.

Name	Thing type
<a href="#">sensor_region1a</a>	-
<a href="#">Sensor1</a>	-

You can choose to create single thing or many things according to your need.

The screenshot shows a step in the AWS IoT 'Create things' wizard. The breadcrumb navigation at the top indicates: AWS IoT > Manage > Things > Create things. The main title 'Create things' has an 'Info' link next to it. A descriptive text explains that a thing resource is a digital representation of a physical device or logical entity in AWS IoT, required for features like Device Shadows, events, jobs, and device management. Below this, a section titled 'Number of things to create' contains two options: 'Create single thing' (selected, highlighted with a blue border) and 'Create many things'. The 'Create single thing' option includes a detailed description of creating a thing resource for a device. At the bottom right of the wizard are 'Cancel' and 'Next' buttons, with 'Next' being highlighted in orange.

AWS IoT > Manage > Things > Create things

## Create things Info

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

### Number of things to create

Create single thing  
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

Create many things  
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Cancel **Next**

Enter a unique name for your device and click ‘next’.

The screenshot shows the 'Specify thing properties' step of the AWS IoT 'Create single thing' wizard. The navigation path is: AWS IoT > Manage > Things > Create things > Create single thing. On the left, there are three optional steps: Step 1 (Specify thing properties), Step 2 - optional (Configure device certificate), and Step 3 - optional (Attach policies to certificate). The main panel is titled 'Specify thing properties' with a 'Info' link. It explains that a thing resource is a digital representation of a physical device or logical entity in AWS IoT. A thing needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features. The 'Thing properties' section contains a 'Thing name' field where 'Sensor\_demo' is entered. Below it is a note: 'Enter a unique name containing only letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.' The 'Additional configurations' section lists several optional configurations: Thing type, Searchable thing attributes, Thing groups, Billing group, and Packages and versions. The 'Device Shadow' section is titled 'Info' and explains that Device Shadows allow connected devices to sync states with AWS. It offers three options: No shadow (selected), Named shadow, and Unnamed shadow (classic). The 'Named shadow' option includes a note: 'Create multiple shadows with different names to manage access to properties, and logically group your devices properties.' At the bottom right are 'Cancel' and 'Next' buttons, with 'Next' being highlighted in orange.

We need to generate a certificate for the thing. Just keep the default settings and click 'Next'.

AWS IoT > Manage > Things > Create things > Create single thing

Step 1  
[Specify thing properties](#)

Step 2 - optional  
**Configure device certificate**

Step 3 - optional  
Attach policies to certificate

## Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

### Device certificate

Auto-generate a new certificate (recommended)  
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

Use my certificate  
Use a certificate signed by your own certificate authority.

Upload CSR  
Register your CA and use your own certificates on one or many devices.

Skip creating a certificate at this time  
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Cancel [Previous](#) [Next](#)

You can create a new policy or attach existing policies to the device certificate for access control. Then, click the ‘Create thing’ button.

The screenshot shows the AWS IoT 'Create single thing' wizard at Step 3: 'Attach policies to certificate - optional'. The left sidebar lists steps: 'Specify thing properties' (Step 1), 'Configure device certificate' (Step 2 - optional), and 'Attach policies to certificate' (Step 3 - optional). The main area title is 'Attach policies to certificate - optional'. It explains that AWS IoT policies grant or deny access to AWS IoT resources, and attaching policies to the device certificate applies this access to the device. A red box highlights the 'Create policy' button. Another red box highlights the list of policies available for selection, which includes 'Sensor1-Policy' and 'Policy2'. The bottom right features 'Cancel', 'Previous', and a large orange 'Create thing' button.

AWS IoT > Manage > Things > Create things > Create single thing

Step 1  
[Specify thing properties](#)

Step 2 - optional  
[Configure device certificate](#)

Step 3 - optional  
**Attach policies to certificate**

**Attach policies to certificate - optional** Info

AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

**Policies (2)**

Select up to 10 policies to attach to this certificate.

[Sensor1-Policy](#)

[Policy2](#)

[Name](#)

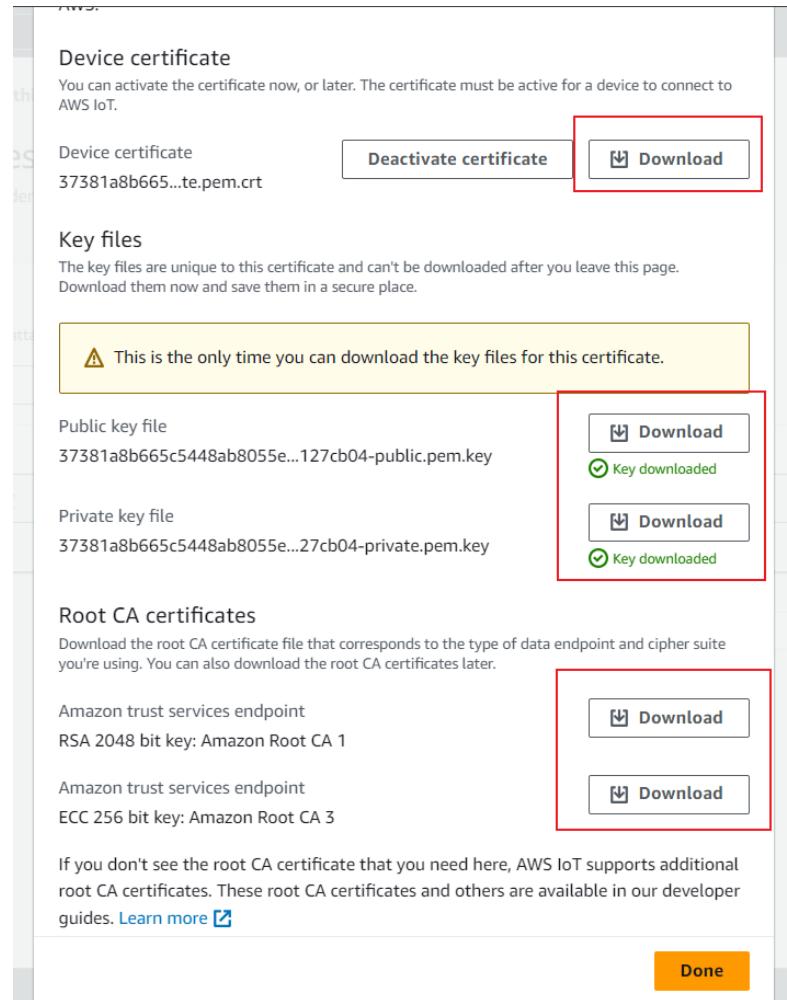
[Create policy](#)

[Filter policies](#)

< 1 > 1

[Cancel](#) [Previous](#) **Create thing**

Download all the certificate and key files to install on your device. You will need them to connect to AWS later.



A new thing has been successfully created.

✓ You successfully created thing Sensor\_demo.

Notifications ⑧ 0 ⑨ 0 ⑩ 2 ⑪ 0 ⑫ 0

[View thing](#) [X](#)

AWS IoT > Manage > Things

**Things (3) Info**

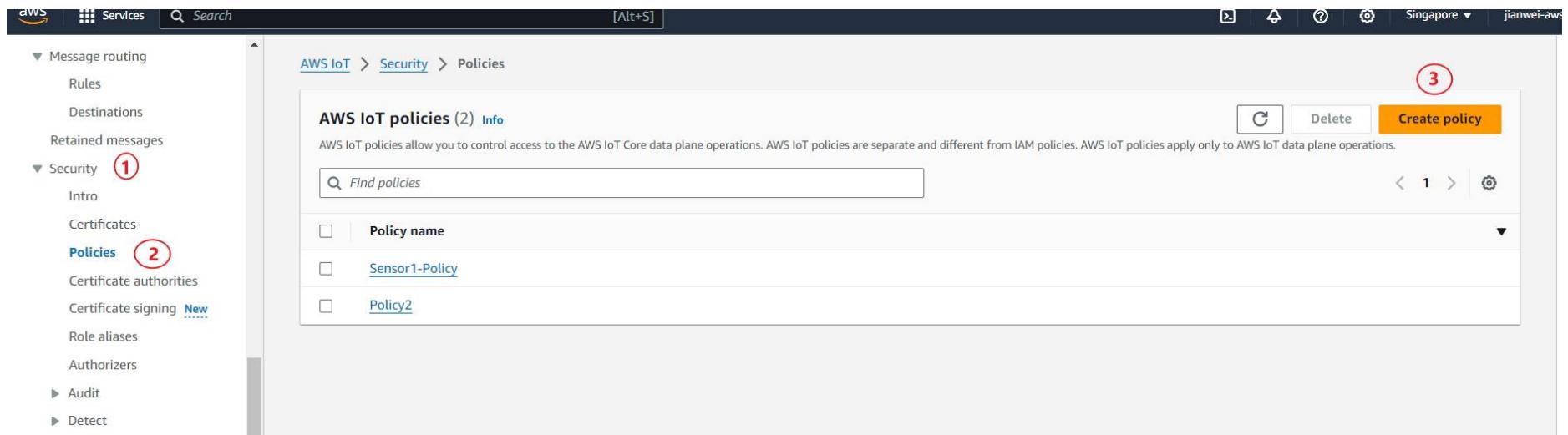
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Filter things by: name, type, group, billing, or searchable attribute.

Name	Thing type
<a href="#">Sensor_demo</a>	-
<a href="#">sensor_region1a</a>	-
<a href="#">Sensor1</a>	-

[Edit](#) [Delete](#) [Create things](#)

To attach a new policy to the created thing, navigate to 'Policies' under the 'Security' section, then click the 'Create policy' button.



1. Enter your policy name.
2. Choose the desired policy actions.
3. Specify your AWS IoT resources, or use '\*' to apply the policy to all resources.

**Create policy** Info

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

**Policy properties**

AWS IoT Core supports named policies so that many identities can reference the same policy document.

**Policy name** 1  
Policy\_demo

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen (-), underscore (\_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

► Tags - optional

**Policy statements** Policy examples

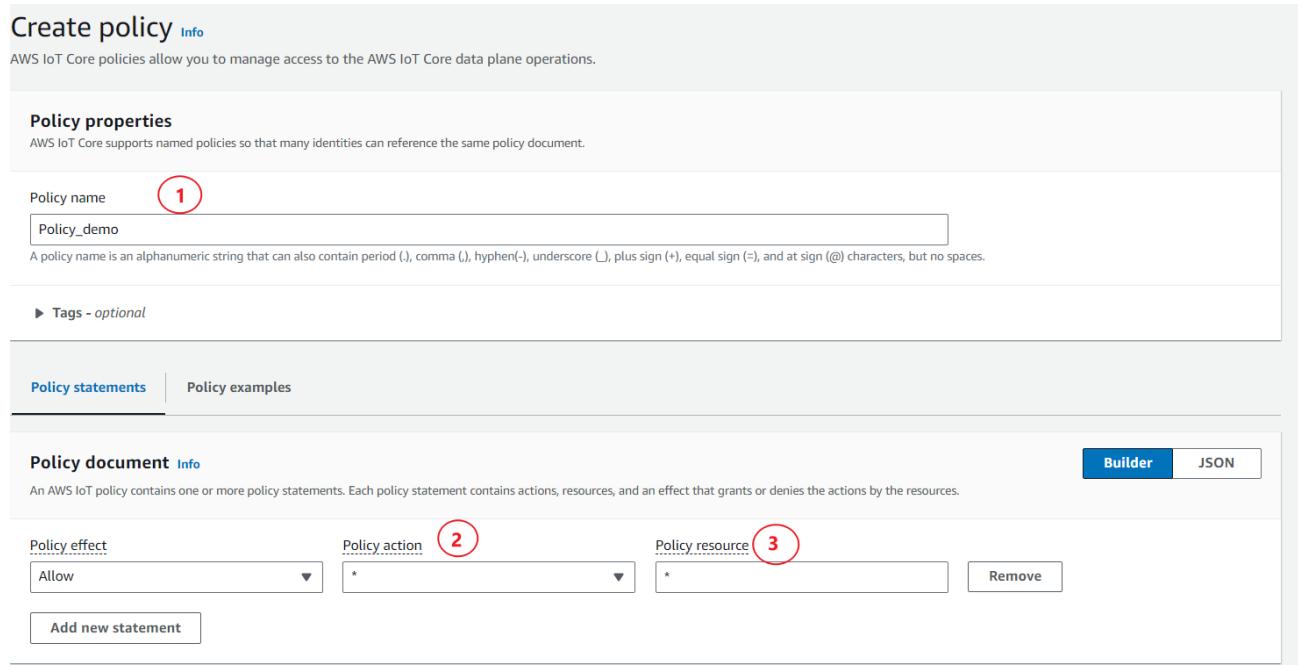
**Policy document** Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

**Builder** **JSON**

**Policy effect** Allow **Policy action** \* 2 **Policy resource** \* 3

Add new statement Remove



A new policy has been successfully created.

⌚ Successfully created policy Policy\_demo.

[View policy](#) [X](#)

[AWS IoT](#) > [Security](#) > Policies

**AWS IoT policies (3) [Info](#)**

AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

[Find policies](#)

[Create policy](#)

< 1 >

<input type="checkbox"/> Policy name
<a href="#">Sensor1-Policy</a>
<a href="#">Policy2</a>
<a href="#">Policy_demo</a>

Now, navigate to the ‘Certificate’ under ‘Security’ section and click on the certificate that you wish to attach the new policy.

The screenshot shows the AWS IoT Certificates page. On the left, a sidebar menu is open, showing various device and security management options. Under the 'Security' section, the 'Certificates' link is highlighted with a blue underline and has a red arrow pointing to it. The main content area displays a table of certificates. The table has columns for 'Certificate ID', 'Status', and 'Created'. There are five entries listed:

Certificate ID	Status	Created
<a href="#">37381a8b665c5448ab8055ea8e7c1312d6cbdd539beb846996070d7ca127cb...</a>	Active	August 17, 2024, 15:47:22 (UTC+08:00)
<a href="#">29b42f47e8673fb03774a92ce2c6b9d89e01462de6bd4a66ce854b576b88482b</a>	Active	July 27, 2024, 17:18:46 (UTC+08:00)
<a href="#">7af6f689afb55a014aae4ebb033e2e2779d8b428da2fb6f71a18dbdec5ec5045</a>	Inactive	July 27, 2024, 15:31:01 (UTC+08:00)
<a href="#">bc6c2311ae4579eaeb66b1ac44c01581cc4580a349251c8f01d55a8f88648917</a>	Active	July 27, 2024, 11:39:46 (UTC+08:00)
<a href="#">d5229e0258c218f4539df05a82832fd0b03a90f24e830f531d210fba0ab76627</a>	Active	July 27, 2024, 11:29:23 (UTC+08:00)

Click ‘Attach policies’.

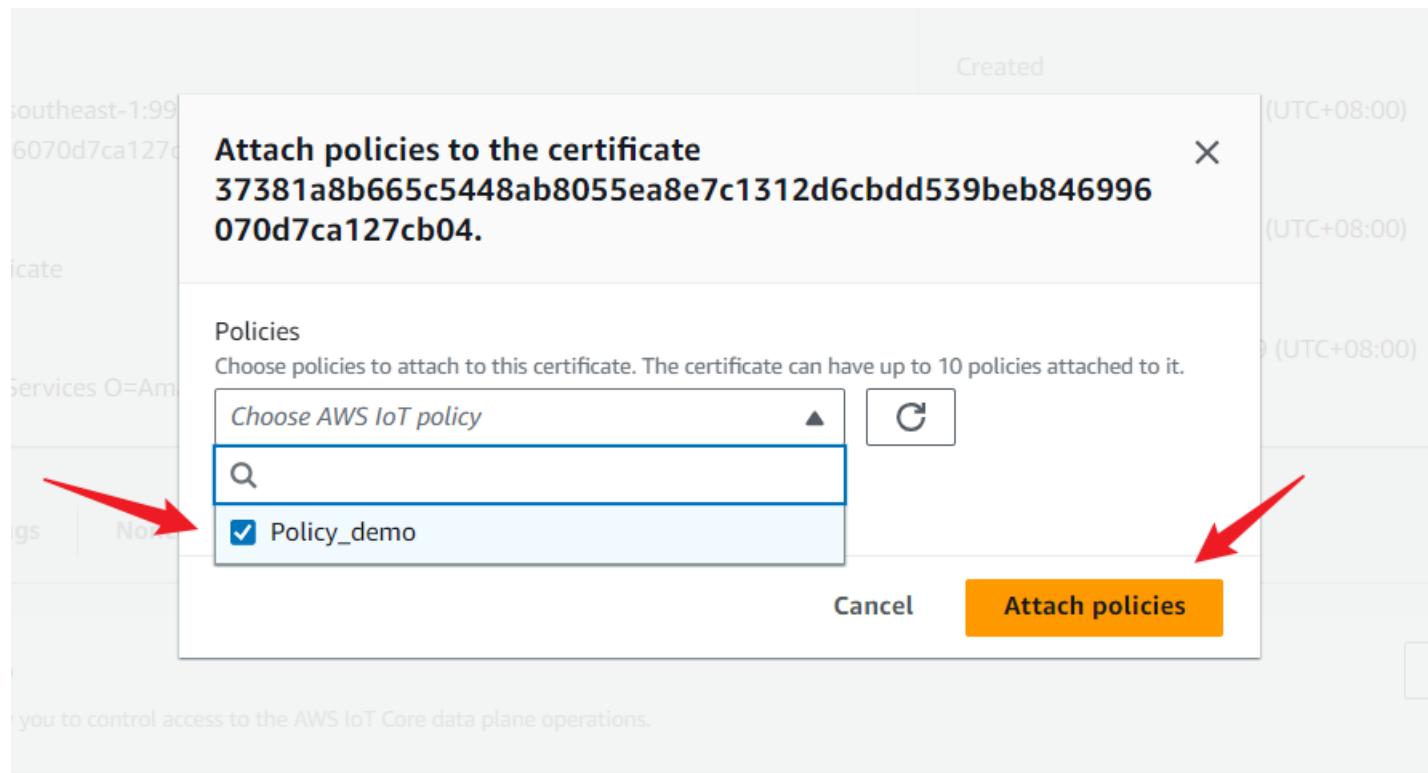
The screenshot shows the AWS IoT Certificates interface. On the left, a sidebar lists various IoT management options like Things, Thing groups, Thing types, Fleet metrics, Greengrass devices, Software packages, Remote actions, Message routing, Rules, Destinations, Retained messages, Security, and Certificates. Under Certificates, Policies and Certificate authorities are listed. Below these are sections for Certificate signing, Role aliases, Authorizers, Audit, Detect, and Fleet Hub.

The main content area displays a success message: "Successfully created policy Policy\_demo." Below this is a "Details" card containing certificate metadata:

Certificate ID	37381a8b665c5448ab8055ea8e7c1312d6cbdd539beb846996070d7ca127cb04
Status	Active
Certificate ARN	arn:aws:iot:ap-southeast-1:992382458892:cert/37381a8b665c5448ab8055ea8e7c1312d6cbdd539beb846996070d7ca127cb04
Created	August 17, 2024, 15:47:22 (UTC+08:00)
Subject	CN=AWS IoT Certificate
Valid	August 17, 2024, 15:45:22 (UTC+08:00)
Issuer	OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US
Expires	January 01, 2050, 07:59:59 (UTC+08:00)

Below the details, there are tabs for Policies, Things, and Noncompliance. The Policies tab is selected, showing a list of policies: Sensor1-Policy and Policy2. To the right of the list are three buttons: a refresh icon, Detach policies, and a highlighted Attach policies button with a red arrow pointing to it.

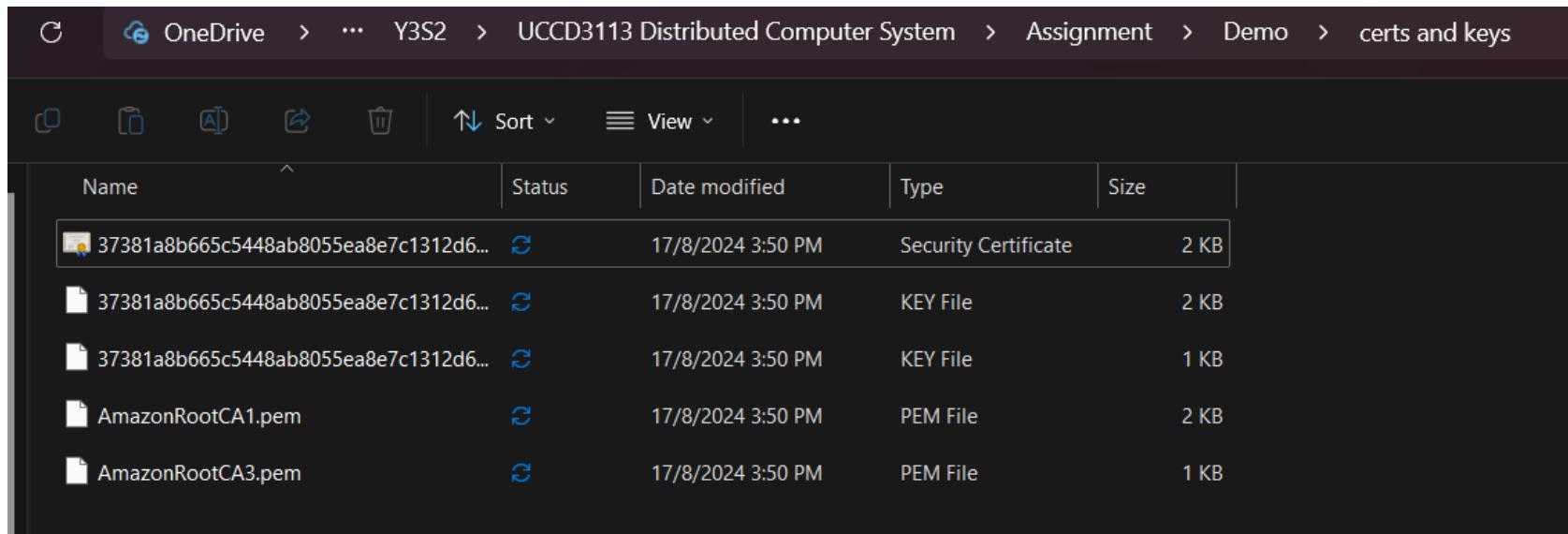
Choose the policy that you created and click ‘Attach policies’.



# AWS IoT SDK

The AWS IoT SDK is a comprehensive toolkit designed to streamline the development, management, and scaling of IoT solutions. It offers features such as device management, data collection, real-time processing, and secure communication through various protocols like MQTT. The SDK supports multiple programming languages, including Python, Java, C++, and JavaScript (Node.js). For more details, refer to this link: <https://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html>. In this manual, we will use Python as our primary language. You can access the device SDK from this link: <https://github.com/aws/aws-iot-device-sdk-python-v2>.

Put all the certificates and keys that downloaded in previous step in one folder.



The screenshot shows a file explorer window with the following path: C:\OneDrive\Y3S2\UCCD3113 Distributed Computer System\Assignment\Demo\certs and keys. The window displays a list of five files:

Name	Status	Date modified	Type	Size
37381a8b665c5448ab8055ea8e7c1312d6...	🕒	17/8/2024 3:50 PM	Security Certificate	2 KB
37381a8b665c5448ab8055ea8e7c1312d6...	🕒	17/8/2024 3:50 PM	KEY File	2 KB
37381a8b665c5448ab8055ea8e7c1312d6...	🕒	17/8/2024 3:50 PM	KEY File	1 KB
AmazonRootCA1.pem	🕒	17/8/2024 3:50 PM	PEM File	2 KB
AmazonRootCA3.pem	🕒	17/8/2024 3:50 PM	PEM File	1 KB

- Imports necessary modules from the AWS IoT SDK and sets up the connection parameters using TLS (certificate and key files).
- Connects to an AWS IoT Core endpoint.

```
from awscrt import io, mqtt, auth, http
from awsiot import mqtt_connection_builder
import time as t
import json
import random
import threading as th
import uuid
```

```
ENDPOINT = "a2dfazm7rcnmac-ats.iot.ap-southeast-1.amazonaws.com"
PATH_TO_CERTIFICATE =
"Demo/37381a8b665c5448ab8055ea8e7c1312d6cbdd539beb846996070d7ca127cb04-certificate.pem.crt"
PATH_TO_PRIVATE_KEY =
"Demo/37381a8b665c5448ab8055ea8e7c1312d6cbdd539beb846996070d7ca127cb04-private.pem.key"
PATH_TO_AMAZON_ROOT_CA_1 = "Demo/AmazonRootCA1.pem"
CLIENT_ID = str(uuid.uuid4())
TOPIC_SUB = "env/volume"
RANGE = 1

received_count = 0

received_all = th.Event()

event_loop_group = io.EventLoopGroup(1)

host_resolver = io.DefaultHostResolver(event_loop_group)

client_bootstrap = io.ClientBootstrap(event_loop_group, host_resolver)

mqtt_connection = mqtt_connection_builder.mtls_from_path(
    endpoint=ENDPOINT, cert_filepath=PATH_TO_CERTIFICATE,pri_key_filepath=PATH_TO_PRIVATE_KEY,
    client_bootstrap=client_bootstrap,ca_filepath=PATH_TO_AMAZON_ROOT_CA_1, client_id=CLIENT_ID,
    clean_session=True, keep_alive_secs=6
)

print(f"Connecting to {ENDPOINT} with client ID '{CLIENT_ID}'...")

connect_future = mqtt_connection.connect()

connect_future.result()

print("Connected!")
```

Subscribes to the MQTT topic env/volume. When a message is received on this topic, a callback function (on\_message\_received) is executed, which prints the message and counts the number of messages received.

```
# Callback when the subscribed topic receives a message

def on_message_received(topic, payload, dup, qos, retain, **kwargs):
    print(f"Message received from topic '{topic}': {payload}")
    global received_count
    received_count += 1
    if received_count == RANGE:
        received_all.set()

    print(f"Subscribing to topic '{TOPIC_SUB}'...")
    subscribe_future, packet_id = mqtt_connection.subscribe(
        topic=TOPIC_SUB,
        qos=mqtt.QoS.AT_LEAST_ONCE,
        callback=on_message_received
    )
    subscribe_result = subscribe_future.result()
    print(f"Subscribed with QoS {subscribe_result['qos']}")
```

```
# Callback when the subscribed topic receives a message

def on_message_received(topic, payload, dup, qos, retain, **kwargs):
    print(f"Message received from topic '{topic}': {payload}")
    global received_count
    received_count += 1
    if received_count == RANGE:
        received_all.set()

    print(f"Subscribing to topic '{TOPIC_SUB}'...")
    subscribe_future, packet_id = mqtt_connection.subscribe(
        topic=TOPIC_SUB,
        qos= mqtt.QoS.AT_LEAST_ONCE,
        callback=on_message_received
    )
    subscribe_result = subscribe_future.result()
    print(f"Subscribed with QoS {subscribe_result['qos']}")
```

Generates random sensor data (soil moisture, temperature, humidity, wind speed) along with a timestamp and a location ID. This data is formatted into a JSON object and published to different MQTT topics (device/1/data, device/2/data, etc.).

```
for i in range(RANGE):
    current_time = t.strftime("%d-%m-%Y %H:%M:%S", t.localtime())
    soil_moisture_level = random.randint(0, 100)
    temperature = random.randint(0, 40)
    humidity = random.randint(0, 100)
    wind_speed = random.randint(0, 12)
    location_id = f"region_{i + 1}"
    TOPIC_PUB = f"device/{i + 1}/data"
    message_pub = {
        "location_id": location_id,
        "soil_moisture_level": soil_moisture_level,
        "temperature": temperature,
        "humidity": humidity,
        "wind_speed": wind_speed,
        "date_time": current_time
    }
    mqtt_connection.publish(
        topic=TOPIC_PUB,
        payload=json.dumps(message_pub),
        qos= mqtt.QoS.AT_LEAST_ONCE
    )
    print(f"Message published to topic '{TOPIC_PUB}':
{json.dumps(message_pub)}")
    t.sleep(1)
```

After receiving all replies, it disconnects from the MQTT broker and ends the program.

```
if not received_all.is_set():

    print("Awaiting reply...")

    received_all.wait()

    print("All replies received")



    print("Disconnecting...")

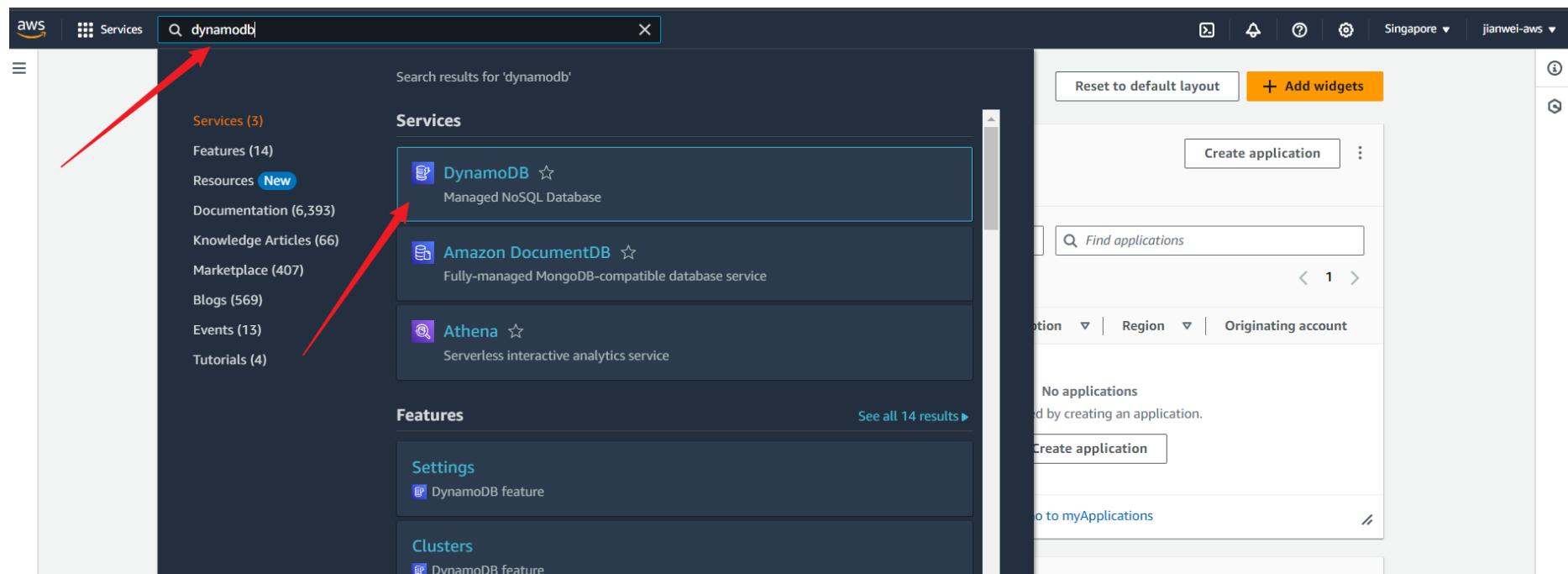
    disconnect_future = mqtt_connection.disconnect()

    disconnect_future.result()

    print("Disconnected. Program complete.")
```

# Amazon DynamoDB

Type “dynamodb” in the search bar and click on the “DynamoDB” service.



Click “Create table”.

The screenshot shows the Amazon DynamoDB Dashboard. On the left, there's a sidebar with links like 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Below that is a 'DAX' section with a dropdown arrow. The main area is titled 'Dashboard' and contains two sections: 'Alarms (0)' and 'DAX clusters (0)'. To the right, there's a 'Create resources' section with a large orange button labeled 'Create table'. A red arrow points from the text in the first instruction to this button. Below it, there's a brief description of the DAX service and another button labeled 'Create DAX cluster'.

DynamoDB

Dashboard

Alarms (0) Info

Find alarms

Manage in CloudWatch

Alarm name

Status

No custom alarms

DAX clusters (0) Info

Find clusters

View details

Cluster name

Status

Create resources

Create an Amazon DynamoDB table for fast and predictable database performance at any scale. [Learn more](#)

Create table

Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. [Learn more](#)

Create DAX cluster

1. Enter 'Table\_demo' as the table name.
2. Enter 'sample\_time' as the Partition key and select **Number** as the data type.
3. Enter 'device\_id' as the Sort key and select **Number** as the data type.
4. At the bottom of the page, click **Create**.

## Create table

**Table details** Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

**Table name**  
This will be used to identify your table.  
**1**  Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.)

**Partition key**  
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.  
**2**   1 to 255 characters and case sensitive.

**Sort key - optional**  
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.  
**3**   1 to 255 characters and case sensitive.

- *sample\_time* serves as the primary key, representing the timestamp when the sample was recorded.
- *device\_id* functions as the sort key, identifying the device that provided the sample.
- *device\_data* contains the data received from the device, formatted by the rule query statement. It will be defined later when configuring the DynamoDB rule action.

Now, we need to create an AWS IoT rule to send data to the DynamoDB table.

# AWS IoT Rule

AWS IoT Rules enable devices to interact with AWS services by processing data sent from IoT devices. These rules listen for specific MQTT messages, apply SQL-based query statements to filter and transform the data, and then route the processed data to various AWS services, such as DynamoDB in our case. First, navigate to the IoT Core service. Then, in the left navigation bar, click on "Rules" under the "Message routing" section, and select "Create rule."

The screenshot shows the AWS IoT Rules management interface. The left sidebar has sections for Monitor, Connect, Test, and Manage. Under Manage, 'Message routing' (marked with a red circle 1) and 'Rules' (marked with a red circle 2) are selected. The main content area displays a table of rules:

Name	Status	Rule topic	Created date
environmental_data_ddb	Inactive	device/+/data	July 28, 2024, 20:25:21 (UTC+08:00)
TriggerLambdaOnData	Inactive	device/+/data	July 29, 2024, 12:38:53 (UTC+08:00)

At the top right, there are buttons for Create rule (highlighted with a red circle 3), Activate, Deactivate, Edit, and Delete.

1. In the Rule name field, enter 'Rule\_demo'.
2. In the Rule description field, provide a detailed description of the rule. A meaningful description will help you easily recall the purpose and function of this rule in the future.
3. Click Next to proceed.

AWS IoT > Message routing > Rules > Create rule

Step 1  
**Specify rule properties**

Step 2  
Configure SQL statement

Step 3  
Attach rule actions

Step 4  
Review and create

Specify rule properties Info

A rule resource contains a list of actions based on the MQTT topic stream.

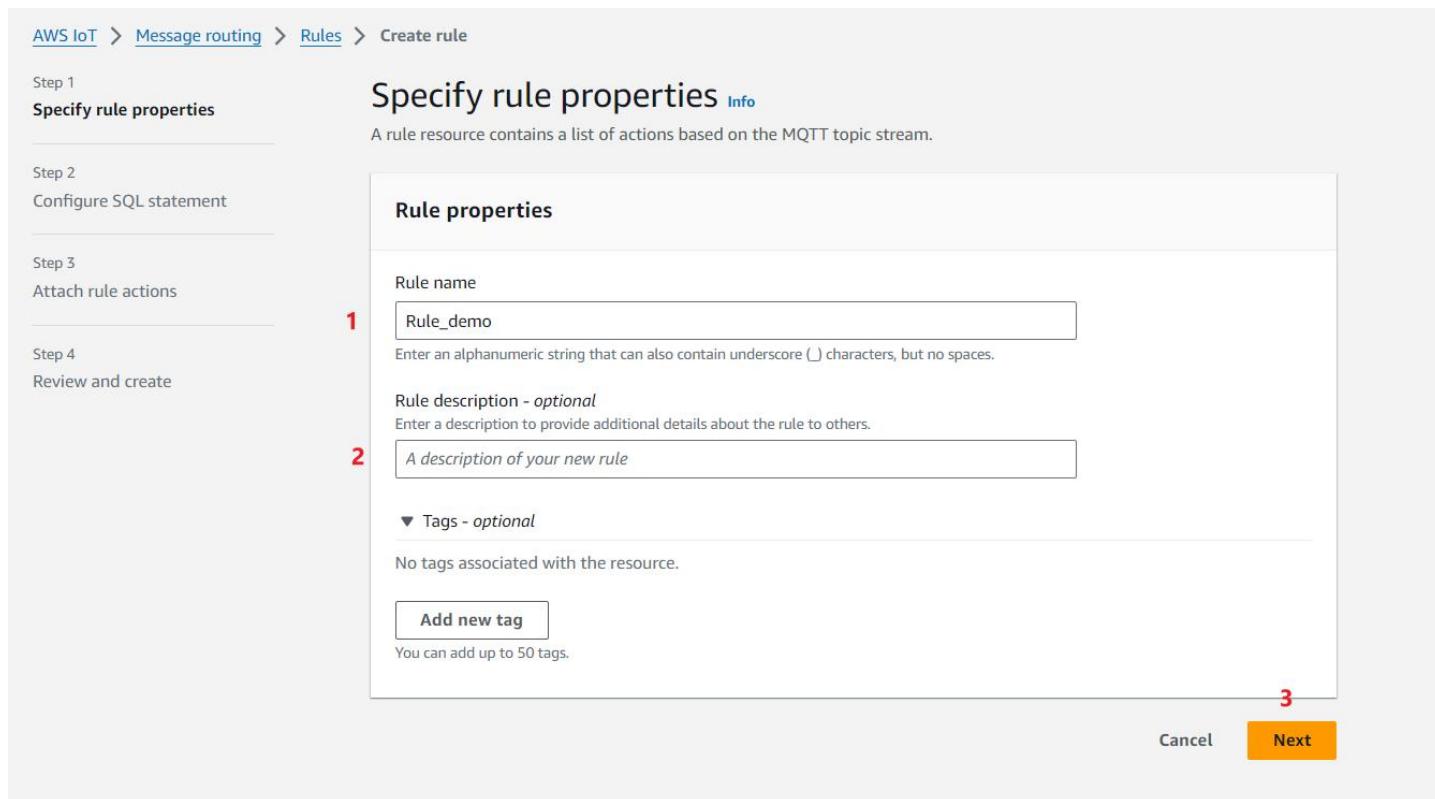
**Rule properties**

1 Rule name  Enter an alphanumeric string that can also contain underscore (\_) characters, but no spaces.

2 Rule description - *optional*  Enter a description to provide additional details about the rule to others.

▼ Tags - *optional*  
No tags associated with the resource.  
 You can add up to 50 tags.

3



1. Select **2016-03-23** in SQL version.
2. Enter the statement in the SQL statement edit box:

```
SELECT temperature, humidity,  
soil_moister_level,date_time,  
wind_speed, location_id  
FROM 'device/+data'
```

## Configure SQL statement Info

Add a simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere.

### SQL statement

#### SQL version

The version of the SQL rules engine to use when evaluating the rule.

1 2016-03-23 ▾

#### SQL statement

Enter a SQL statement using the following: `SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>`. For example: `SELECT temperature FROM 'iot/topic' WHERE temperature > 50`. To learn more, see [AWS IoT SQL Reference](#).

2 1 `SELECT temperature, humidity, soil_moister_level,date_time,`  
2   `wind_speed, location_id`  
3 `FROM 'device/+data'`

The SQL statement listens for MQTT messages that match the device/+data topic filter and passes all the attributes unchanged.

1. In Rule actions, choose **DynamoDB** in Action1.
2. In Table name, choose the name of the DynamoDB table you created in a previous step: **Table\_demo**.

The screenshot shows the 'Create rule' process in the AWS IoT Rules interface. The current step is 'Attach rule actions'. The top section, 'SQL statement', contains the following query:

```
SELECT temperature, humidity, soil_moister_level,date_time,  
wind_speed, location_id  
FROM 'device/+/data'
```

The bottom section, 'Rule actions', shows 'Action 1' configured as follows:

- Action type: **DynamoDB** (selected from a dropdown menu)
- Table name: **Table\_demo** (selected from a dropdown menu)
- Buttons: 'Remove' (next to the table name dropdown), 'View' (with a magnifying glass icon), and 'Create DynamoDB table' (with a plus icon).

1. For the Partition key, enter sample\_time.
2. For the Partition key value, use \${timestamp()}. This will substitute the value returned by the timestamp function instead of using a value from the message payload.
3. For the Sort key, enter device\_id.
4. For the Sort key value, use \${cast(topic(2) AS DECIMAL)}. This substitution template will take the second element from the topic name (the device's ID), cast it to a DECIMAL value, and insert it to match the numeric format of the key.
5. In the "Write message data to this column" field, enter device\_data. This will create a device\_data column in the DynamoDB table.
6. Leave the Operation field blank.

**Partition key**

The partition key (also called hash key) must match the partition key of the DynamoDB table that you created.

1 sample\_time

**Partition key type**

The partition key (also called hash key) type can be STRING or NUMBER. The default value is STRING.

NUMBER

**Partition key value**

The partition key (also called hash key) value supports substitution templates that provide data at runtime.

2 \${timestamp()}

**Sort key - optional**

The sort key (also called range key) must match the sort key of the DynamoDB table that you created.

3 device\_id

**Sort key type**

The sort key (also called range key) type can be STRING or NUMBER. The default value is STRING.

NUMBER

**Sort key value**

The sort key (also called range key) value supports substitution templates that provide data at runtime.

4 \${cast(topic(2) AS DECIMAL)}

**Write message data to this column - optional**

5 device\_data

**Operation - optional**

The operation can be INSERT, UPDATE or DELETE. The default value is INSERT.

6 |INSERT

1. In IAM role, choose Create new role.
2. In the Create role dialog box, for Role name, enter ***demo\_ddb\_role***. This new role will automatically contain a policy with a prefix of "aws-iot-rule" that will allow the ***Rule\_demo*** rule to send data to the ***Table\_demo*** DynamoDB table you created.
3. In IAM role, choose **wx\_ddb\_role**.
4. At the bottom of the page, choose Next.

The screenshot shows the second step of the AWS IoT Rule Creation Wizard. It has two main sections: 'Operation - optional' and 'IAM role'.

**Operation - optional:** A text input field containing 'INSERT'. Below it is a note: 'The operation can be INSERT, UPDATE or DELETE. The default value is INSERT.'

**IAM role:** A dropdown menu showing 'demo\_ddb\_role'. To its right are three buttons: 'View' with a magnifying glass icon, a 'C' icon, and a 'Create new role' button. The 'Create new role' button is highlighted with a red oval. Below the dropdown is a note: 'AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.'

**Add rule action** (button)

**Error action - optional:** A note: 'You can optionally set an action that will be executed when something goes wrong with processing your rule. If two rule actions in the same rule fail, the error action receives one message that contains both errors.' Below it is a 'Add error action' button.

At the bottom are three buttons: 'Cancel', 'Previous', and 'Next'. The 'Next' button is highlighted with a red oval.

At the bottom of the "Review and create" page, click "Create" to finalize and create the rule. You will see that a new ***Rule\_demo*** has been successfully created.

The screenshot shows the AWS IoT Rules list page. At the top, a green banner displays a success message: "Successfully created role demo\_ddb\_role". Below the banner, there is a notifications bar with icons for notifications, alarms, and other alerts. The main header includes the navigation path: AWS IoT > Message routing > Rules, and a "Create rule" button. The table below lists three rules:

Name	Status	Rule topic	Created date
<a href="#">environmental_data_ddb</a>	Inactive	device/+/data	July 28, 2024, 20:25:21 (UTC+08:00)
<a href="#">Rule_demo</a>	Active	device/+/data	August 22, 2024, 16:56:48 (UTC+08:00)
<a href="#">TriggerLambdaOnData</a>	Inactive	device/+/data	July 29, 2024, 12:38:53 (UTC+08:00)

Next, we will use the MQTT client to publish and subscribe to the MQTT messages for testing the new rule.

1. Click on the MQTT test client in the navigation bar.
2. Enter *device/+/data* as the input topic filter.
3. Click ‘Subscribe’.

The screenshot shows the AWS IoT console with the MQTT test client selected in the navigation bar. A success message at the top indicates a role was created. The main area displays the MQTT test client interface with three numbered steps:

- Step 1:** The 'MQTT test client' link in the navigation bar is highlighted with a red circle.
- Step 2:** The 'Topic filter' input field contains the value 'device/+data', which is highlighted with a red circle.
- Step 3:** The 'Subscribe' button is highlighted with a red circle.

The interface includes sections for 'Connection details', 'Subscribe to a topic' (with the active topic filter), and 'Publish to a topic'.

1. Choose ***Publish to a topic***.
2. Enter the input topic with a specific device ID, ***device/18/data***.
3. Enter the following sample data as ***Message payload***.

```
{  
  "temperature": 28,  
  "humidity": 80,  
  "soil_moister_level": 1013,  
  "wind_speed": 5,  
  "location_id": "region_1",  
  "date_time": "28-07-2024 20:25:54"  
}
```

4. Choose ***Publish***

Subscribe to a topic

**Publish to a topic ①**

Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

device/18/data ②

X

Message payload ③

```
{  
  "temperature": 28,  
  "humidity": 80,  
  "soil_moister_level": 1013,  
  "wind_speed": 5,  
  "location_id": "region_1",  
  "date_time": "28-07-2024 20:25:54"  
}
```

► Additional configuration

**Publish**

④

You will see that a new record has been inserted into DynamoDB successfully.

The screenshot shows the AWS DynamoDB console interface. On the left, the navigation pane includes options like Dashboard, Tables, Explore items (which is selected), PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, Settings, Clusters, Subnet groups, Parameter groups, and Events. A large red arrow points from the bottom-left towards the 'Items returned' section.

The main workspace displays the 'Table\_demo' table details. Under 'Scan or query items', the 'Scan' button is selected. The 'Select a table or index' dropdown is set to 'Table - Table\_demo' and the 'Select attribute projection' dropdown is set to 'All attributes'. Below these are 'Filters' and 'Run' buttons. A green success message at the bottom states: 'Completed. Read capacity units consumed: 0.5'.

In the 'Items returned (1)' section, there is one item listed:

	sample_time (Number)	device_id (Number)	device_data
<a href="#">1724319761440</a>	18	{ "temperature": { "N": "28" }, "soil_moister_level": { "N": "1013" }, "hum...	

Next, when a new record is inserted into DynamoDB, we need to calculate the water volume based on the newly inserted environmental data and send the result back to the IoT device to trigger the irrigation instruction. This can be achieved using AWS Lambda, a serverless compute service that runs code without the need to provision or manage servers.

# AWS Lambda

1. Enter **Lambda** in search bar and choose the **Lambda** service.
2. Click **Create function**.

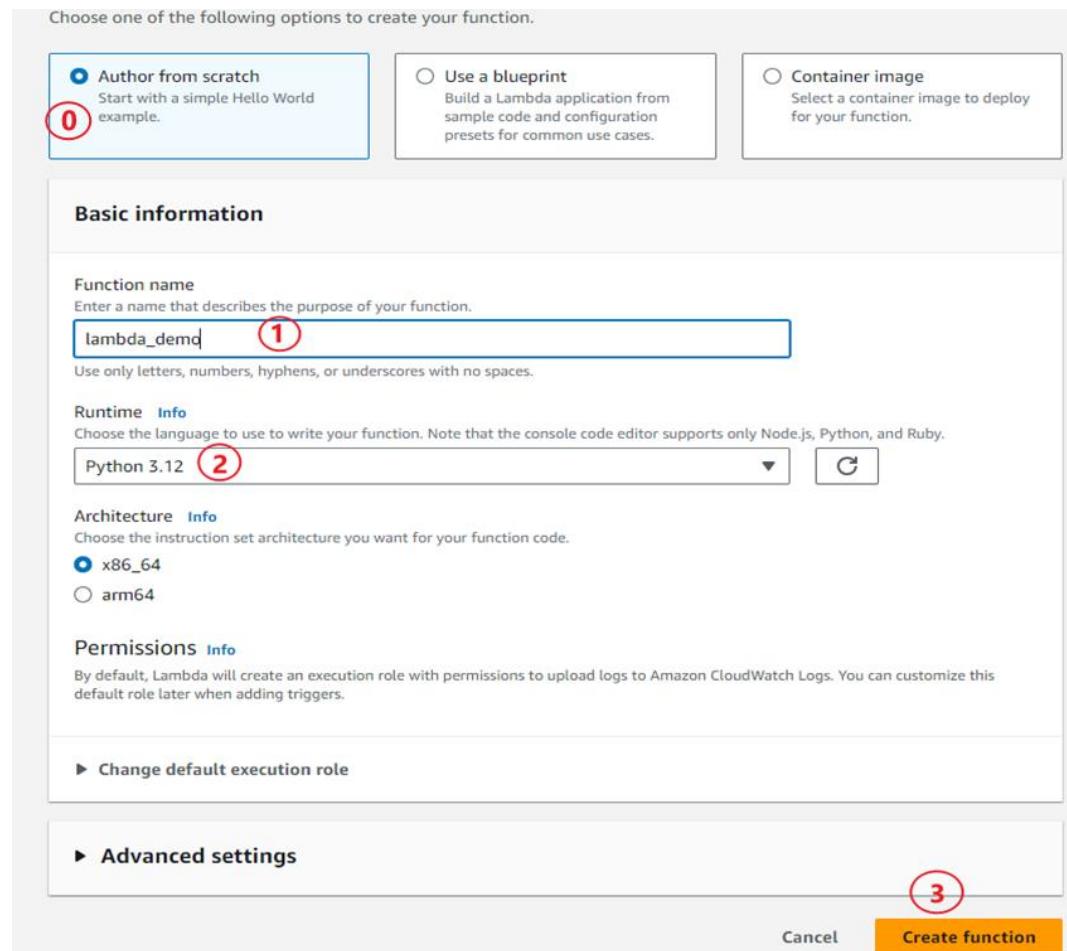
The screenshot shows the AWS Lambda service interface. The top navigation bar includes the AWS logo, Services dropdown, a search bar with the text "lambda" (circled with red number 1), and other navigation icons. The main left sidebar has a "Lambda" heading and sections for Dashboard, Applications, Functions (which is selected and circled with red number 1), Additional resources, and Related AWS resources. The main content area is titled "Functions (1)" and shows a table with one item: "CalculateWaterVolume" (Function name), "-", (Description), Zip (Package type), Python 3.8 (Runtime), and 17 days ago (Last modified). A "Create function" button is located at the top right of the table area (circled with red number 2).

Function name	Description	Package type	Runtime	Last modified
<a href="#">CalculateWaterVolume</a>	-	Zip	Python 3.8	17 days ago

0. Choose *Author from scratch*.

1. Enter *lambda\_demo* as the function name.

2. Select *Python 3.12* for the Runtime.



After the function has been successfully created, click on ‘Add trigger’.

The screenshot shows the AWS Lambda Functions overview for a function named 'lambda\_demo'. A green success message at the top states: 'Successfully created the function lambda\_demo. You can now change its code and configuration. To invoke your function with a test event, choose "Test".' Below the message, the function name 'lambda\_demo' is displayed with a 'Diagram' tab selected. On the right side, there are several buttons: 'Throttle', 'Copy ARN', 'Actions ▾', 'Export to Application Composer', and 'Download ▾'. The 'Function ARN' field shows the value: 'arn:aws:lambda:ap-southeast-1:992382458892:function:lambda\_demo'. A red arrow points to the '+ Add trigger' button located on the left side of the main content area.

1. Select **DynamoDB** as the source.
2. Choose the **Table\_demo** that you previously created.
3. Enter **1** for the batch size, meaning that the Lambda function will be triggered every time a single new record is inserted.
4. Select **Latest** for the starting position. This means that the Lambda function will process only the records added after the function is created.
5. Leave the Batch window blank.
6. Click **Add**

## Add trigger

**Trigger configuration [Info](#)**

**1**  **DynamoDB**  
aws database event-source-mapping nosql polling

**2**

**Activate trigger**  
Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).

**3** **Batch size**  
The number of records in each batch to send to the function.

**4** **Starting position**  
The position in the stream to start reading from. For more information, see [ShardIteratorType](#) in the Amazon Kinesis API Reference.

**5** **Batch window - optional**  
The maximum amount of time to gather records before invoking the function, in seconds.

**6**

In order to read from the DynamoDB trigger, your execution role must have proper permissions.

1. The trigger **Table\_demo** was successfully added to function **lambda\_demo**.
2. Scroll down to the **code source** section to edit the code.

lambda\_demo

Throttle Copy ARN Actions ▾

Function overview Info Export to Application Composer Download ▾

Diagram Template

lambda\_demo (0)

DynamoDB (1) + Add destination

+ Add trigger

Description -

Last modified 14 minutes ago

Function ARN arn:aws:lambda:ap-southeast-1:992382458892:function:lambda\_demo

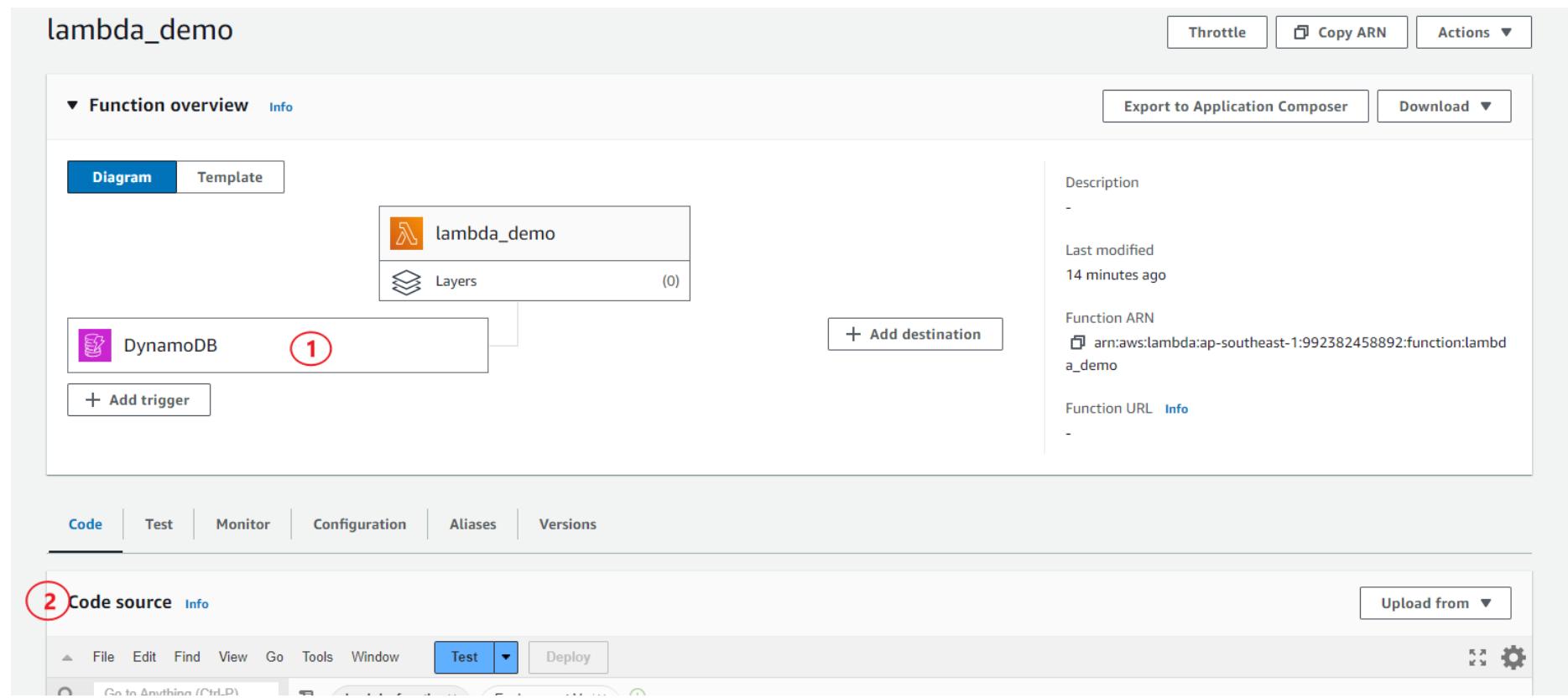
Function URL Info -

Code Test Monitor Configuration Aliases Versions

Code source Info Upload from ▾

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl+D)



Enter the following code into the lambda\_function.py

```
import json
import boto3
import logging
import csv
import io

# Initialize logger
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# Initialize AWS IoT Data Client
iot_client = boto3.client('iot-data', region_name='ap-southeast-1')

# Initialize S3 Client
s3_client = boto3.client('s3')
bucket_name = 'yourbucketname'
csv_key = 'data/water_usage.csv'
```

```
def lambda_handler(event, context):
    logger.info("Received event: %s", json.dumps(event))

    # Check if 'Records' key is in the event
    if 'Records' not in event:
        logger.error("Invalid event structure: 'Records' key missing")
        return {
            'statusCode': 400,
            'body': json.dumps('Invalid event structure')
        }

    for record in event['Records']:
        logger.info("Processing record: %s", json.dumps(record))
```

```
if record['eventName'] == 'INSERT':  
  
    new_image = record['dynamodb']['NewImage']  
  
    device_data = new_image['device_data']['M']  
  
    logger.info("New image: %s", json.dumps(new_image))  
  
    # Extract the environmental data  
  
    humidity = float(device_data['humidity']['N'])  
  
    temperature = float(device_data['temperature']['N'])  
  
    wind_speed = float(device_data['wind_speed']['N'])  
  
    soil_moisture = float(device_data['soil_moister_level']['N'])  
  
    location_id = device_data['location_id']['S']  
  
    date_time = device_data['date_time']['S']  
  
    # Calculate the water volume (example calculation)  
  
    water_volume = calculate_water_volume(humidity,  
                                           temperature, wind_speed, soil_moisture)  
  
    logger.info("Calculated water volume: %s", water_volume)
```

```
# Prepare the new row to append to the CSV  
  
new_row = [humidity, temperature,  
          wind_speed, soil_moisture, location_id, date_time,  
          water_volume]  
  
# Read the existing CSV from S3  
  
try:  
  
    csv_obj =  
    s3_client.get_object(Bucket=bucket_name,  
                         Key=csv_key)  
  
    csv_body =  
    csv_obj['Body'].read().decode('utf-8')  
  
    csv_reader =  
    csv.reader(io.StringIO(csv_body))  
  
    csv_data = list(csv_reader)  
  
except s3_client.exceptions.NoSuchKey:
```

```
# If the CSV file doesn't exist, create a new one with headers

csv_data = [["humidity", "temperature", "wind_speed",
"soil_moisture", "location_id", "date_time", "water_volume"]]

# Append the new row to the CSV data

csv_data.append(new_row)

# Convert the CSV data back to string

csv_buffer = io.StringIO()

csv_writer = csv.writer(csv_buffer)

csv_writer.writerows(csv_data)

csv_str = csv_buffer.getvalue()

# Store the updated CSV back to S3

s3_client.put_object(
    Bucket=bucket_name,
    Key=csv_key,
    Body=csv_str
)
```

```
logger.info("Updated CSV stored in S3: %s",
csv_key)

# Publish the water volume to the IoT topic

topic = 'env/volume'

message = {

    'water_volume': water_volume,
    'location_id': location_id,
    'date_time': date_time
}

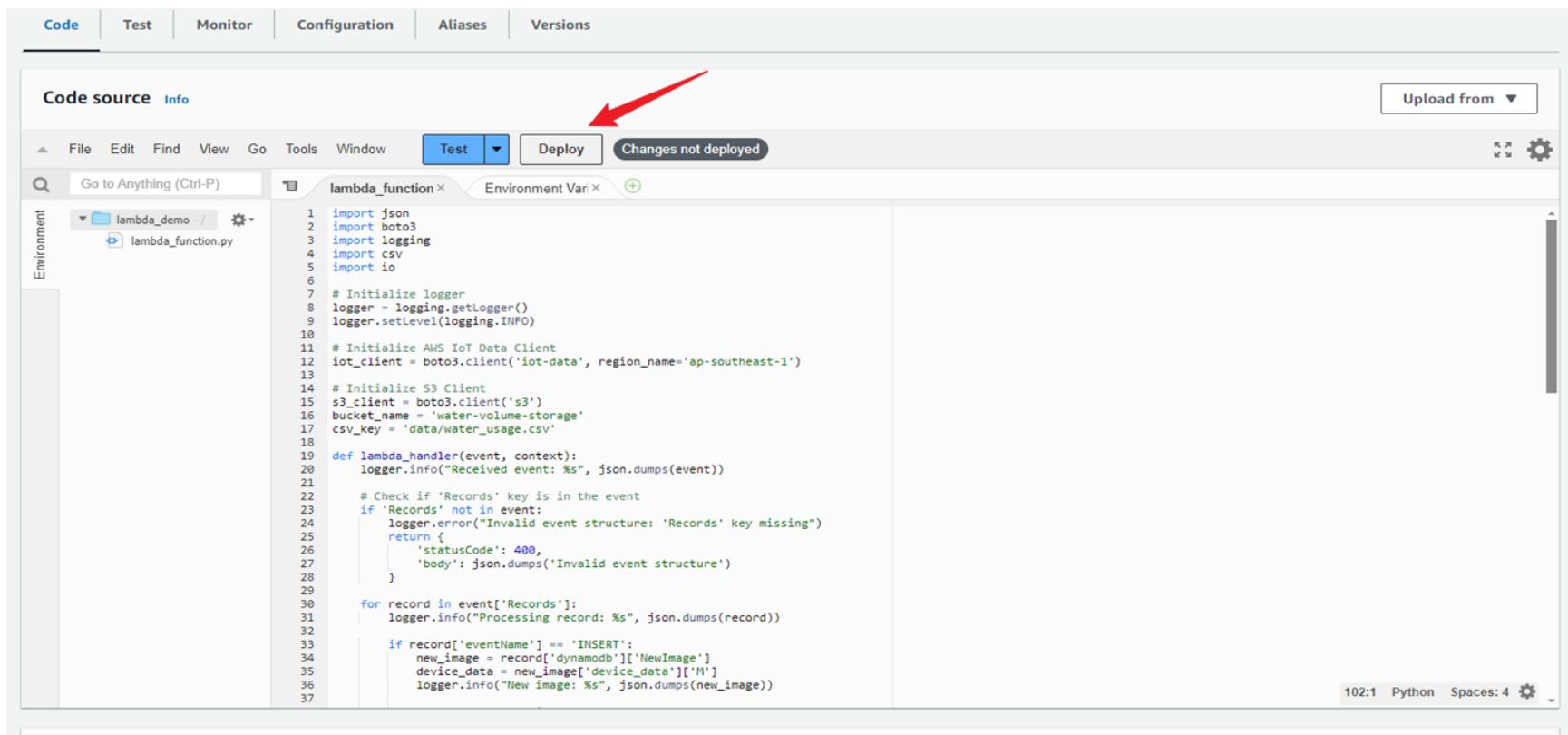
response = iot_client.publish(
    topic=topic,
    qos=1,
    payload=json.dumps(message)
)

logger.info("Publish response: %s", response)

return {
    'message': 'Success'
}
```

```
def calculate_water_volume(humidity, temperature, wind_speed, soil_moisture):
    # Example calculation logic
    water_volume = (100 - soil_moisture) * (1 + humidity / 100) * (1 - wind_speed / 10) * temperature / 30
    return water_volume
```

This code is a Lambda function that processes environmental data inserted into a DynamoDB table, calculates water volume, updates a CSV file in an S3 bucket, and publishes the result to an AWS IoT topic. Click the **Deploy** button to save changes and deploy the code.



The screenshot shows the AWS Lambda console interface. At the top, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Code tab is selected. Below the tabs, there are buttons for Test, Deploy, and a status message 'Changes not deployed'. A red arrow points to the Deploy button. The main area contains a code editor with Python code for a Lambda function named lambda\_function. The code imports json, boto3, logging, csv, and io. It initializes a logger, AWS IoT Data Client, and S3 Client. It defines a lambda\_handler function that checks for the 'Records' key in the event. If missing, it returns an error. Otherwise, it processes each record in the 'Records' array. For an 'INSERT' event, it extracts 'dynamodb' and 'device\_data' from the record, creates a new image object, and updates a CSV file in an S3 bucket. The code editor has syntax highlighting and line numbers. At the bottom right, there are status indicators: 102:1 Python Spaces: 4.

```
1 import json
2 import boto3
3 import logging
4 import csv
5 import io
6
7 # Initialize logger
8 logger = logging.getLogger()
9 logger.setLevel(logging.INFO)
10
11 # Initialize AWS IoT Data Client
12 iot_client = boto3.client('iot-data', region_name='ap-southeast-1')
13
14 # Initialize S3 Client
15 s3_client = boto3.client('s3')
16 bucket_name = 'water-volume-storage'
17 csv_key = 'data/water_usage.csv'
18
19 def lambda_handler(event, context):
20     logger.info("Received event: %s", json.dumps(event))
21
22     # Check if 'Records' key is in the event
23     if 'Records' not in event:
24         logger.error("Invalid event structure: 'Records' key missing")
25         return {
26             'statusCode': 400,
27             'body': json.dumps('Invalid event structure')
28         }
29
30     for record in event['Records']:
31         logger.info("Processing record: %s", json.dumps(record))
32
33         if record['eventName'] == 'INSERT':
34             new_image = record['dynamodb']['NewImage']
35             device_data = new_image['device_data']['M']
36             logger.info("New image: %s", json.dumps(new_image))
37
```

At this point, we can test the SDK program created in the previous step. If, upon running the program, you receive a reply with the water volume, it means all the services are working correctly. Congratulations!

```
C:\Users\Oct23\OneDrive - Universiti Tunku Abdul Rahman\Degree\Y3S2\UCCD3113 Distributed Computer System\Assignment\cert>test.py
Connecting to a2dfazm7rcnmac-ats.iot.ap-southeast-1.amazonaws.com with client ID 'b60cbaa7-b545-4b18-b52d-1c74d0b1bc7e'...
Connected!
Subscribing to topic 'env/volume'...
Subscribed with QoS 1
Published message to topic 'device/1/data': {"location_id": "region_1", "soil_moister_level": 65, "temperature": 15, "humidity": 80, "wind_speed": 3, "date_time": "28-08-2024 21:16:41"}
Received message from topic 'env/volume': b'{"water_volume": 22.04999999999997, "location_id": "region_1", "date_time": "28-08-2024 21:16:41"}'
All replies received
Disconnecting...
Disconnected! Program complete.
```

# Amazon S3

Now, we need to create a bucket to store the water usage volume. First, navigate to the S3 service by typing ‘S3’ in the search bar. Then, click on ***Create bucket***.

The screenshot shows the AWS S3 service page. At the top, there is a navigation bar with the AWS logo, a 'Services' dropdown, a search bar containing 'Search [Alt+S]', and a user profile icon for 'jianwei'. A red circle labeled '①' is positioned above the search bar. Below the navigation bar, the main title is 'Amazon S3' with a close button 'X'. On the left, a sidebar menu includes 'Buckets', 'Access Grants', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'IAM Access Analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens' (with 'Dashboards', 'Storage Lens groups', and 'AWS Organizations settings' sub-options), and a 'Feature spotlight' section. A red circle labeled '②' is positioned above the 'Create bucket' button. The main content area is titled 'Amazon S3' and features an 'Account snapshot - updated every 24 hours' section with a link to 'All AWS Regions'. It also includes a 'View Storage Lens dashboard' button. Below this, there are tabs for 'General purpose buckets' (which is selected) and 'Directory buckets'. The 'General purpose buckets' section shows a table with one item:

Name	AWS Region	IAM Access Analyzer	Creation date
<a href="#">water-volume-storage</a>	Asia Pacific (Singapore) ap-southeast-1	<a href="#">View analyzer for ap-southeast-1</a>	July 29, 2024, 16:53:43 (UTC+08:00)

Below the table are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket' (which is highlighted with a red circle).

Enter ***bucketdemounique*** as the bucket name, leave the remaining settings as default, then scroll down to the bottom of the page and click ***Create bucket***.

The screenshot shows the 'Create bucket' wizard in the Amazon S3 console. The top navigation bar indicates the user is at 'Amazon S3 > Buckets > Create bucket'. The main title is 'Create bucket' with an 'Info' link. Below it, a sub-instruction says 'Buckets are containers for data stored in S3.'

**General configuration**

- AWS Region: Asia Pacific (Singapore) ap-southeast-1
- Bucket name:  A red arrow points to this field.
- Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)
- Copy settings from existing bucket - *optional*: Only the bucket settings in the following configuration are copied.
  - [Choose bucket](#)
- Format: s3://bucket/prefix

**Object Ownership** [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

**ACLs disabled (recommended)**  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

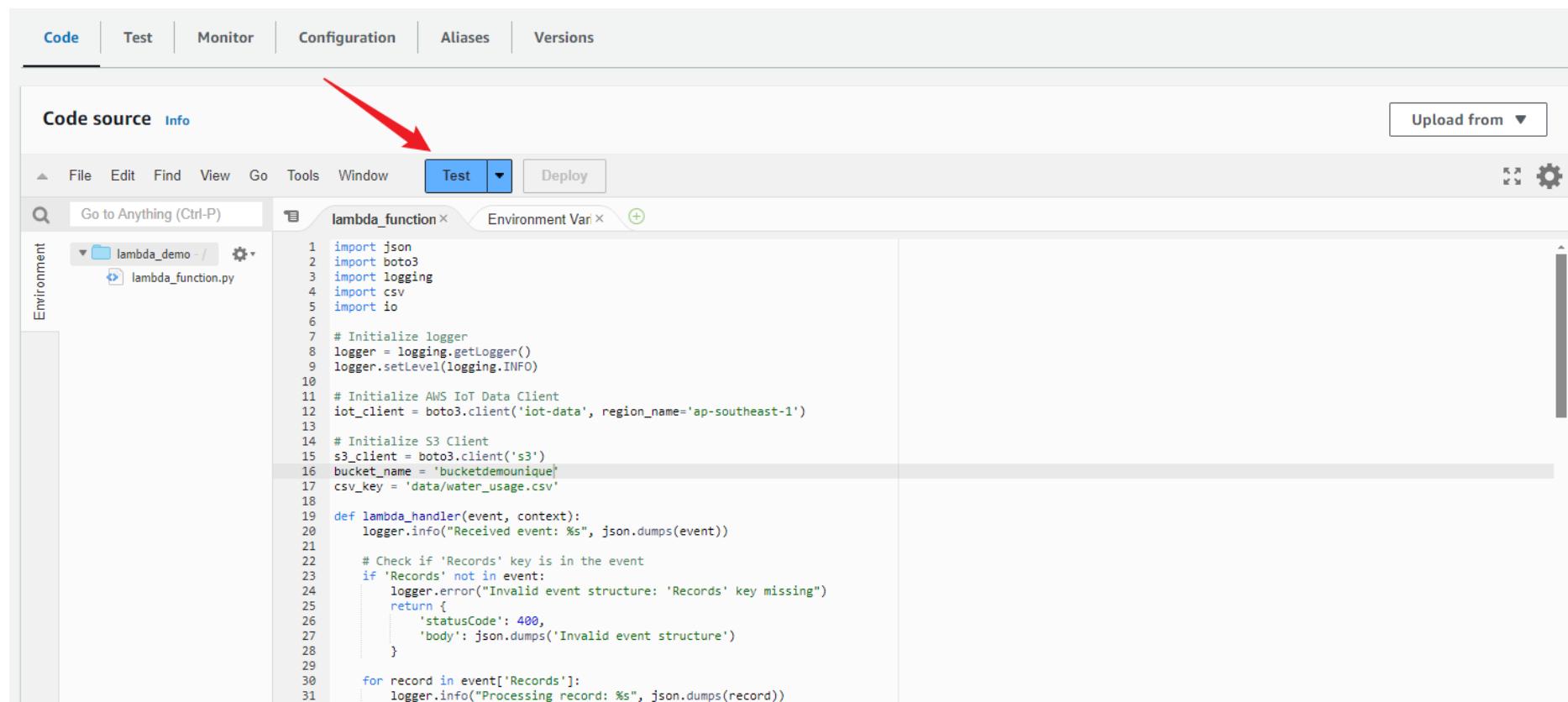
**ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership  
Bucket owner enforced

After creating the bucket, you can now upload files to it. Return to the Lambda function you previously created and replace the bucket name with your newly created '*bucketdemounique*' bucket.

```
# Initialize S3 Client  
  
s3_client = boto3.client('s3')  
  
bucket_name = 'bucketdemounique'  
  
csv_key = 'data/water_usage.csv'
```

To make sure everything is working correctly, go back to the lambda function and click on ***Test*** button.



The screenshot shows the AWS Lambda code editor interface. At the top, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Code tab is selected. Below the tabs is a toolbar with File, Edit, Find, View, Go, Tools, Window, a Test button (which has a red arrow pointing to it), Deploy, and a gear icon. The main area shows a file named lambda\_function.py with the following code:

```
1 import json
2 import boto3
3 import logging
4 import csv
5 import io
6
7 # Initialize logger
8 logger = logging.getLogger()
9 logger.setLevel(logging.INFO)
10
11 # Initialize AWS IoT Data Client
12 iot_client = boto3.client('iot-data', region_name='ap-southeast-1')
13
14 # Initialize S3 Client
15 s3_client = boto3.client('s3')
16 bucket_name = 'bucketdemounique'
17 csv_key = 'data/water_usage.csv'
18
19 def lambda_handler(event, context):
20     logger.info("Received event: %s", json.dumps(event))
21
22     # Check if 'Records' key is in the event
23     if 'Records' not in event:
24         logger.error("Invalid event structure: 'Records' key missing")
25         return {
26             'statusCode': 400,
27             'body': json.dumps('Invalid event structure')
28         }
29
30     for record in event['Records']:
31         logger.info("Processing record: %s", json.dumps(record))
```

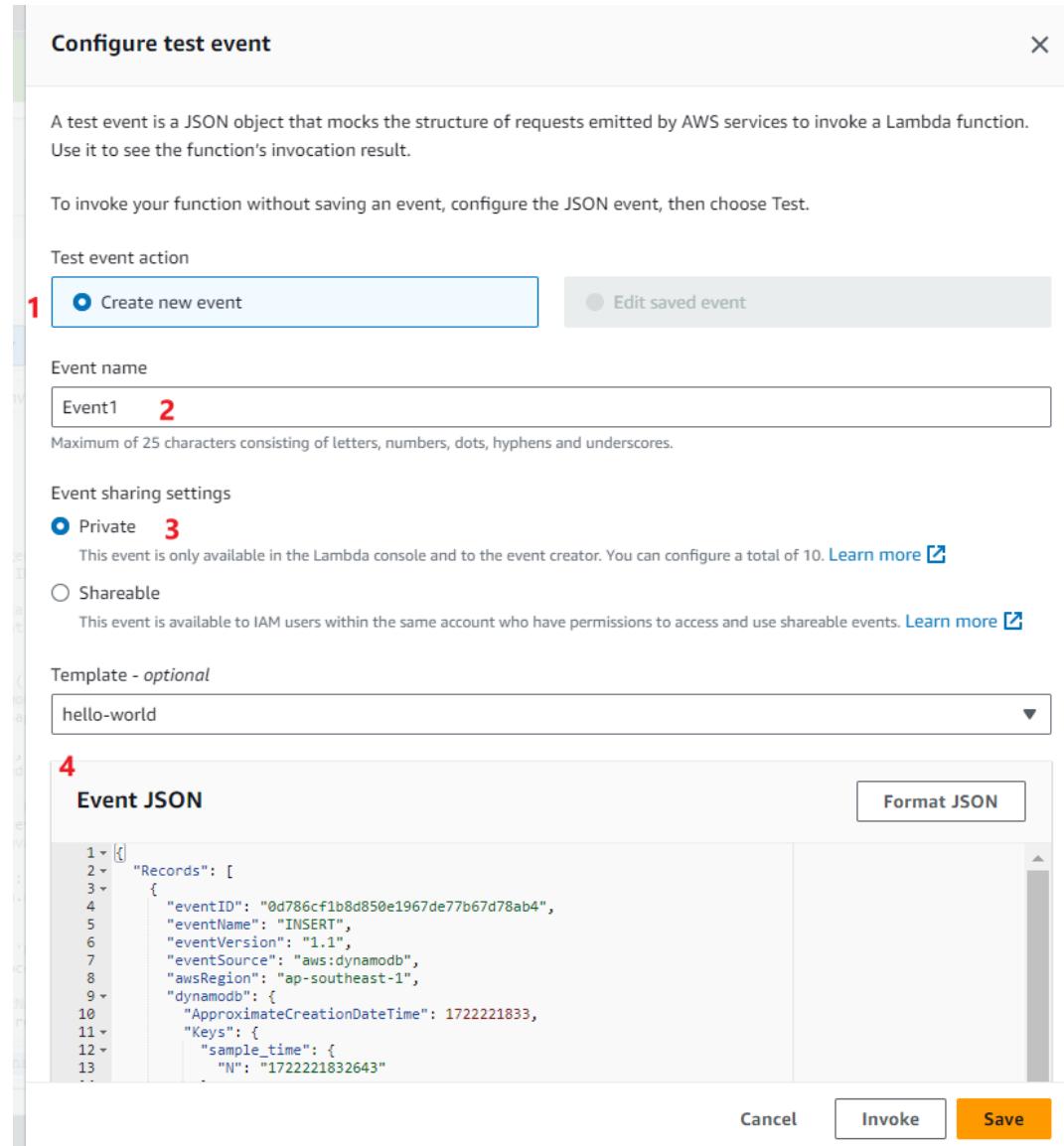
1. Select ‘Create new event’.
2. Enter ‘Event1’ as the event name.
3. Select ‘Private’ for the event sharing settings.
4. Enter the following event JSON:

```
{  
  "Records": [  
    {  
      "eventID": "0d786cf1b8d850e1967de77b67d78ab4",  
      "eventName": "INSERT",  
      "eventVersion": "1.1",  
      "eventSource": "aws:dynamodb",  
      "awsRegion": "ap-southeast-1",  
      "dynamodb": {  
        "ApproximateCreationDateTime": 1722221833,  
        "Keys": {  
          "sample_time": {  
            "N": "1722221832643"  
          }  
        }  
      }  
    }  
  ]  
}
```

```
"NewImage": {  
  "sample_time": {  
    "N": "1722221832643"  
  },  
  "device_data": {  
    "M": {  
      "soil_moister_level": {  
        "N": "34"  
      }  
    },  
    "date_time": {  
      "S": "29-07-2024 10:57:10"  
    },  
  }  
}
```

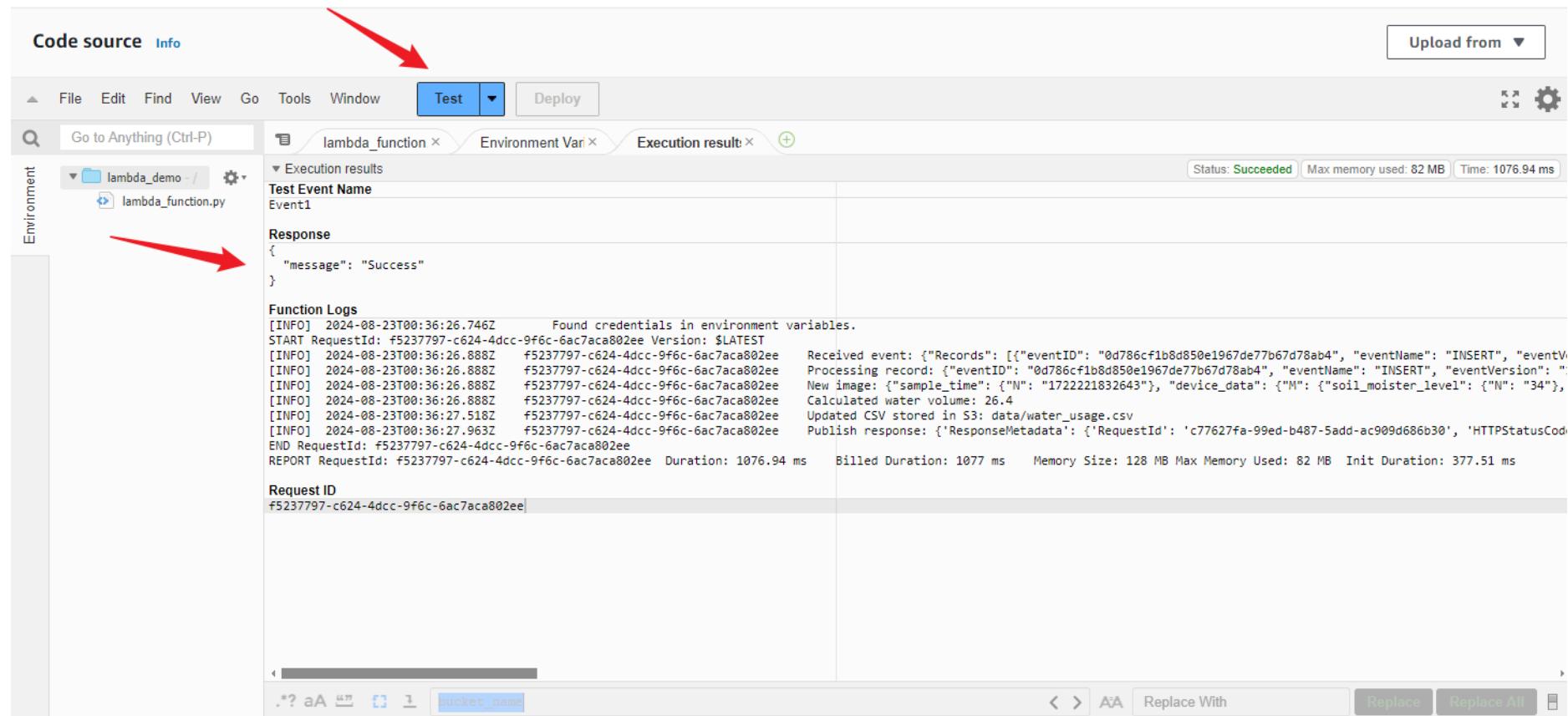
```
"temperature": {  
    "N": "12"  
},  
"location_id": {  
    "S": "region_2"  
}  
}  
},  
"device_id": {  
    "N": "1"  
}  
},  
"humidity": {  
    "N": "100"  
},  
"wind_speed": {  
    "N": "5"  
},
```

```
"SequenceNumber": "841320000000036341113576",  
"SizeBytes": 182,  
"StreamViewType": "NEW_AND_OLD_IMAGES"  
},  
"eventSourceARN": "arn:aws:dynamodb:ap-southeast-  
1:992382458892:table/environmental_data/stream/2024-07-  
28T09:02:46.192"  
}  
]  
}
```

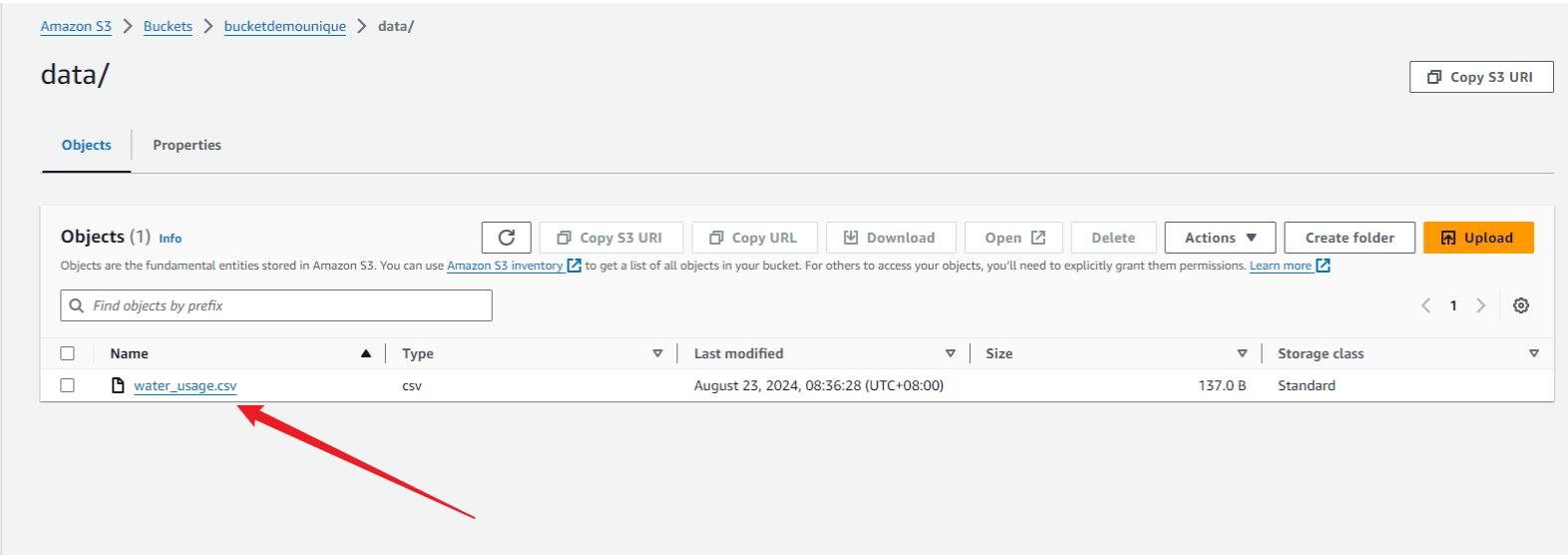


Click on **Save** button.

Click on Test button and you should see a response with “Success” message.



Go back to your ***bucketdemounique*** bucket. You should see that a new ***water\_usage.csv*** file has been inserted into the bucket.



The screenshot shows the Amazon S3 console interface. On the left, there is a navigation sidebar with various links such as Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens (which is expanded to show Dashboards, Storage Lens groups, and AWS Organizations settings), and Feature spotlight. The main area displays the contents of the 'data/' folder within the 'bucketdemounique' bucket. The 'Objects' tab is selected. At the top of the object list, there are several actions: Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload. Below these are search and pagination controls. A table lists the single object: Name (water\_usage.csv), Type (csv), Last modified (August 23, 2024, 08:36:28 (UTC+08:00)), Size (137.0 B), and Storage class (Standard). A red arrow points to the 'water\_usage.csv' link in the Name column. The URL for the object is [https://s3.console.aws.amazon.com/s3/buckets/bucketdemounique/data/water\\_usage.csv?region=ap-southeast-1&tab=objects](https://s3.console.aws.amazon.com/s3/buckets/bucketdemounique/data/water_usage.csv?region=ap-southeast-1&tab=objects).

Name	Type	Last modified	Size	Storage class
<a href="https://s3.console.aws.amazon.com/s3/buckets/bucketdemounique/data/water_usage.csv?region=ap-southeast-1&amp;tab=objects">water_usage.csv</a>	csv	August 23, 2024, 08:36:28 (UTC+08:00)	137.0 B	Standard

# Amazon SageMaker

With all the data stored in the S3 bucket, we can use it to forecast future water usage demand by using Amazon SageMaker, a fully managed machine learning (ML) service where you can build, train, and deploy ML models at scale. To use the service, type **SageMaker** in search bar and click the *Amazon SageMaker* service.

The screenshot shows the AWS search interface with the search term 'sageMaker' entered in the search bar (marked with a red circle 1). The search results page for 'sageMaker' is displayed, with the 'Services' section showing one result: 'Amazon SageMaker' (marked with a red circle 2). The result card for Amazon SageMaker includes the service icon, the name, a star icon, and the tagline 'Build, Train, and Deploy Machine Learning Models'. Below the services section, there are other categories like Features, Resources, Documentation, Knowledge Articles, Marketplace, Blogs, Events, and Tutorials, each with their respective icons and descriptions.

Click ***notebooks*** from the navigation bar.

The screenshot shows the Amazon SageMaker console interface. On the left, there is a navigation sidebar with the following structure:

- Getting started
- Applications and IDEs
  - Studio
  - Canvas
  - RStudio
  - Notebooks (this item is circled with a red arrow)
- Admin configurations
  - Domains
  - Role manager
  - Images
  - Lifecycle configurations
- SageMaker dashboard
- Search

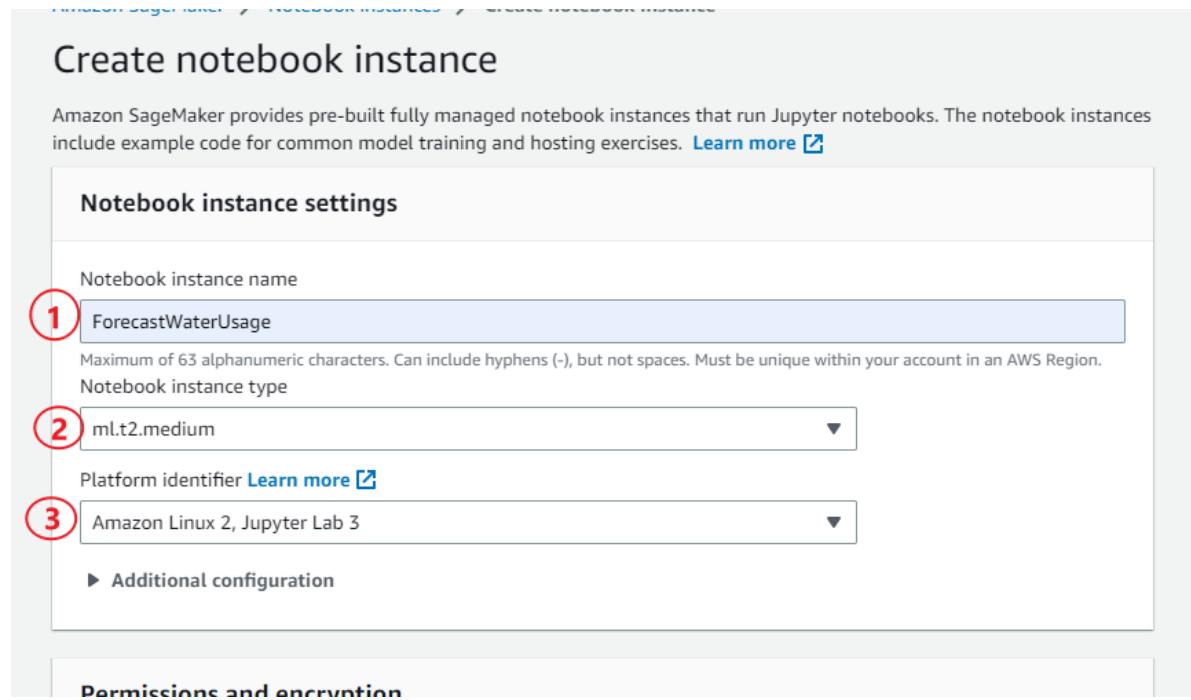
The main content area displays the "Domains" page. At the top, there is a banner about the "Automatic UI upgrade". Below the banner, the breadcrumb navigation shows "Amazon SageMaker > Domains". The page title is "Domains (0) Info". A search bar labeled "Find domain name" is present. A table header with columns "Name", "Id", "Status", "Created on", and "Modified on" is shown, followed by a message "No domains" and the instruction "To add a domain, choose Create domain." Action buttons "View" and "Create domain" are located at the top right of the table area.

Click *Create notebook instance*.

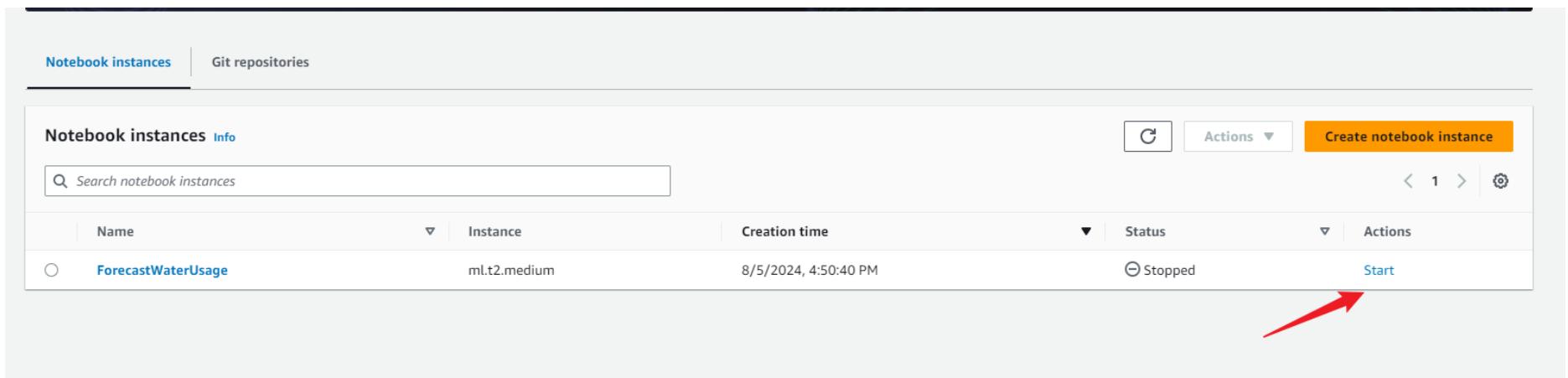
The screenshot shows the Amazon SageMaker interface. On the left, there's a sidebar with various navigation options like 'Getting started', 'Applications and IDEs' (which is expanded to show 'Studio', 'Canvas', 'RStudio', and 'Notebooks'), 'Admin configurations' (expanded to show 'Domains', 'Role manager', 'Images', and 'Lifecycle configurations'), 'JumpStart' (expanded to show 'Foundation models', 'Computer vision models', and 'Natural language processing models'), and general links for 'SageMaker dashboard' and 'Search'. The main content area is titled 'Notebooks and Git repos' and contains a section titled 'Try the new JupyterLab in SageMaker Studio'. This section includes a 'Get Started' button and a link to 'How to access JupyterLab in Studio?'. Below this is a table titled 'Notebook instances' with one entry: 'ForecastWaterUsage' (Name), 'ml.t2.medium' (Instance), '8/5/2024, 4:50:40 PM' (Creation time), 'Stopped' (Status), and a 'Start' button (Actions). A prominent red arrow points to the 'Create notebook instance' button at the top right of the table.

Name	Instance	Creation time	Status	Actions
ForecastWaterUsage	ml.t2.medium	8/5/2024, 4:50:40 PM	Stopped	<a href="#">Start</a>

1. Enter '***ForecastWaterUsage***' as notebook instance name.
2. Select '***ml.t2.medium***' as notebook instance type.
3. Select '***Amazon Linux 2, Jupyter Lab 3***' as platform identifier.
4. Leave other settings as default, scroll down to the bottom of the page, and click ***Create notebook instance***.



Once your notebook has been created, click on *Start*.



The screenshot shows the 'Notebook instances' tab selected in a dashboard. The page title is 'Notebook instances Info'. There is a search bar labeled 'Search notebook instances'. On the right, there are buttons for 'Actions' (with a dropdown arrow) and 'Create notebook instance'. Below these are navigation icons for back, forward, and refresh. The main content is a table with the following columns: Name, Instance, Creation time, Status, and Actions. The table contains one row for a notebook named 'ForecastWaterUsage' with the details: Instance 'ml.t2.medium', Creation time '8/5/2024, 4:50:40 PM', Status 'Stopped', and Actions including a 'Start' button. A red arrow points to the 'Start' button.

Name	Instance	Creation time	Status	Actions
ForecastWaterUsage	ml.t2.medium	8/5/2024, 4:50:40 PM	Stopped	<a href="#">Start</a>

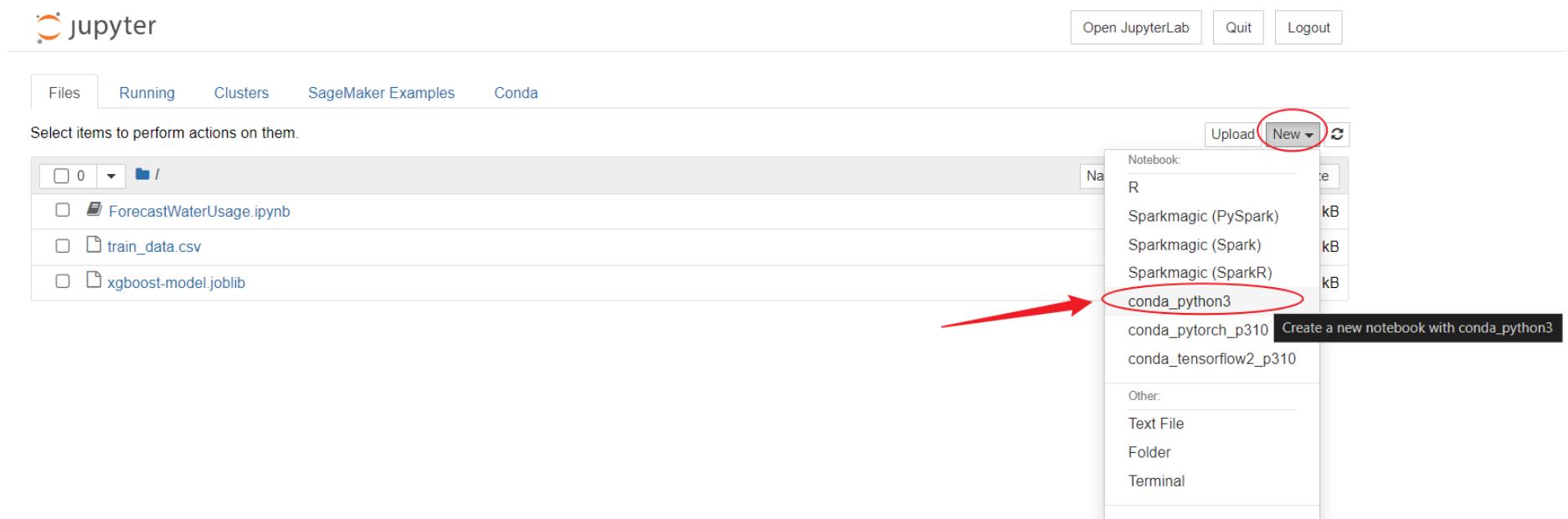
Click ***Open Jupyter***.

The screenshot shows a user interface for managing notebook instances. At the top, there are two tabs: "Notebook instances" (which is selected) and "Git repositories". Below the tabs is a search bar labeled "Search notebook instances". The main area displays a table of notebook instances with the following columns: Name, Instance, Creation time, Status, and Actions. There is one entry in the table:

Name	Instance	Creation time	Status	Actions
ForecastWaterUsage	ml.t2.medium	8/5/2024, 4:50:40 PM	InService	<a href="#">Open Jupyter</a>   <a href="#">Open JupyterLab</a>

A red arrow points to the "Open Jupyter" link in the Actions column of the first row.

You will be redirected to a new Jupyter page. Click on *New* and select *conda\_python3*.



Now, we want to write a script to reads historical water usage data from S3, trains an ARIMA model to predict future usage, generates a forecast CSV, uploads it to S3, and visualizes the results. Enter the following code into your notebook:

*Import all the necessary libraries.*

```
import pandas as pd  
from statsmodels.tsa.arima.model import ARIMA  
import matplotlib.pyplot as plt  
import boto3  
import io
```

*Specifies the S3 bucket and the location of the historical water usage data (water\_usage.csv) and reads the csv data from S3 into a pandas DataFrame.*

```
bucket = 'water-volume-storage'

data_key = 'data/water_usage.csv'

data_location = f's3://{{bucket}}/{{data_key}}'

# Load the data

df = pd.read_csv(data_location, parse_dates=['date_time'], index_col='date_time', dayfirst=True)

# Ensure water_volume is numeric

df['water_volume'] = pd.to_numeric(df['water_volume'], errors='coerce')

# Convert index to DatetimeIndex, allowing for mixed formats

if not isinstance(df.index, pd.DatetimeIndex):

    df.index = pd.to_datetime(df.index, format='mixed', dayfirst=True)

# Aggregate data (assuming daily water volume usage)

df = df.resample('D').sum().interpolate(method='linear')
```

*Trains the ARIMA model on the historical water\_volume data.*

```
# Train the ARIMA model

model = ARIMA(df['water_volume'], order=(5, 1, 0)) # Example order, can be optimized

model_fit = model.fit()

# Forecast future water usage

forecast_steps = 30 # Predict for the next 30 days

forecast = model_fit.forecast(steps=forecast_steps)

# Prepare the forecast data for CSV

start_date = df.index[-1].strftime('%Y-%m-%d')

end_date = pd.date_range(df.index[-1], periods=forecast_steps + 1, freq='D')[-1].strftime('%Y-%m-%d')

forecast_filename = f'forecast_total_water_usage_{start_date}_to_{end_date}.csv'

# Create a DataFrame for the forecast

forecast_dates = pd.date_range(df.index[-1], periods=forecast_steps + 1, freq='D')[1:]

forecast_df = pd.DataFrame({'date': forecast_dates, 'forecasted_water_volume': forecast})

total_forecasted_usage = forecast.sum()
```

*Uploads the forecast CSV to the S3 bucket in a forecast directory.*

```
# Add total forecasted usage as a final row

total_row = pd.DataFrame({'date': ['Total'], 'forecasted_water_volume':
[total_forecasted_usage]})

forecast_df = pd.concat([forecast_df, total_row])

# Convert DataFrame to CSV format in memory

csv_buffer = io.StringIO()

forecast_df.to_csv(csv_buffer, index=False)

# Upload the CSV file to S3

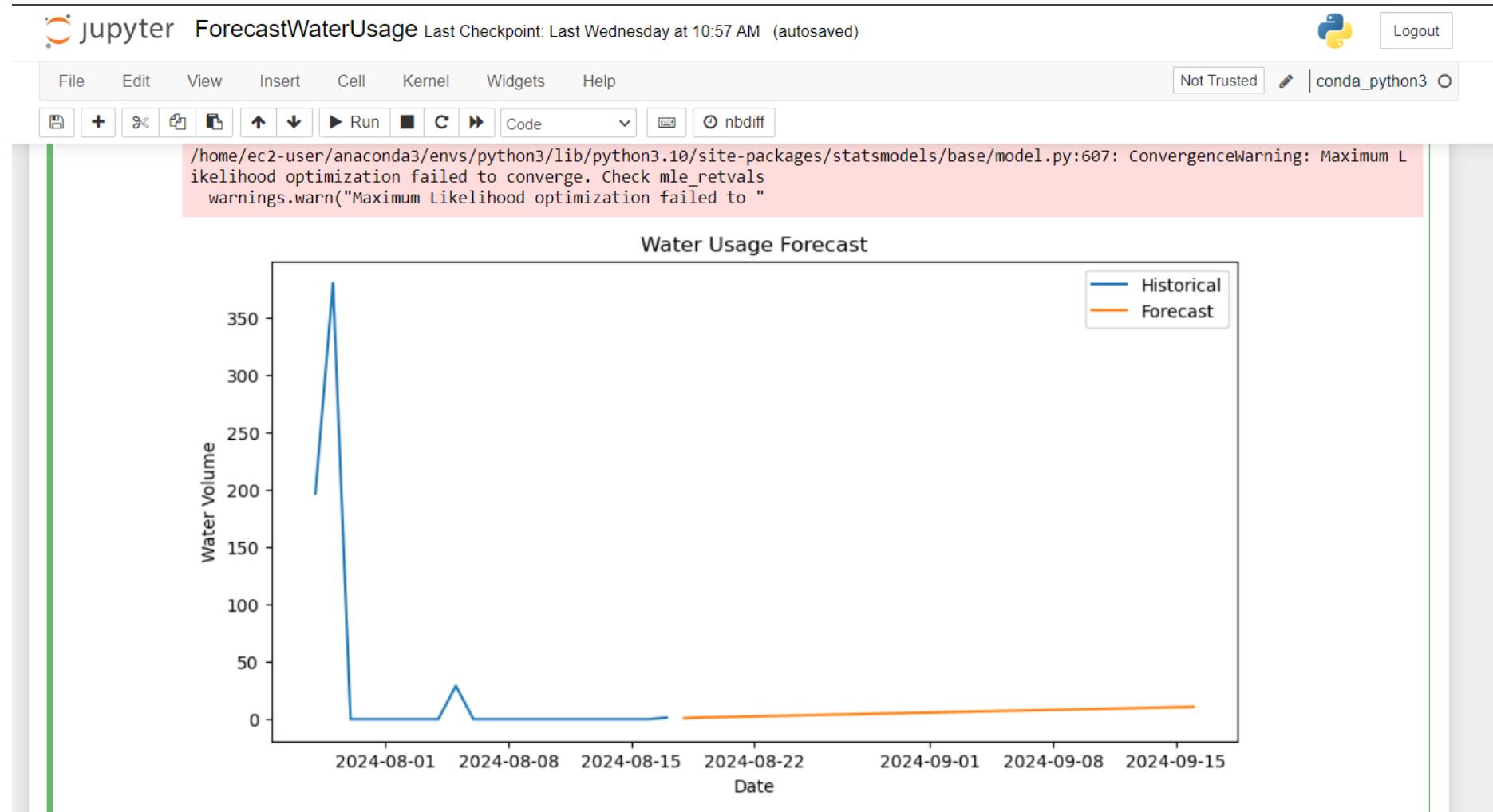
s3_client = boto3.client('s3')

s3_client.put_object(Body=csv_buffer.getvalue(), Bucket=bucket,
Key=f'forecast/{forecast_filename}')
```

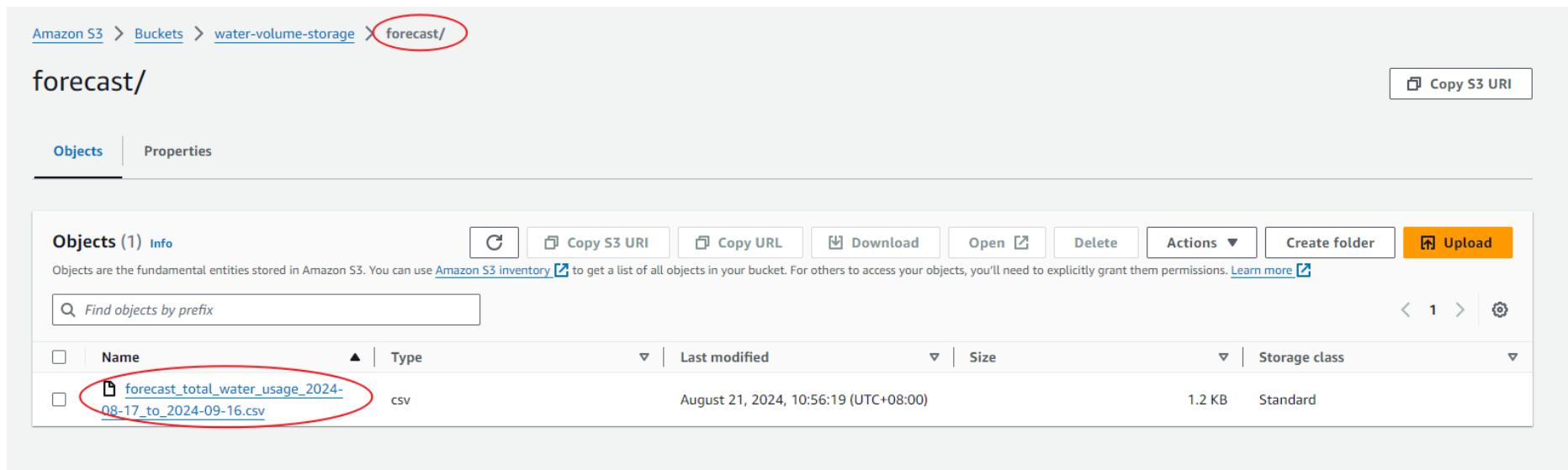
*Plots the historical water usage and the forecasted values on a graph.*

```
# Plot the results  
  
plt.figure(figsize=(10, 5))  
  
plt.plot(df.index, df['water_volume'], label='Historical')  
  
plt.plot(forecast_dates, forecast, label='Forecast')  
  
plt.xlabel('Date')  
  
plt.ylabel('Water Volume')  
  
plt.title('Water Usage Forecast')  
  
plt.legend()  
  
plt.show()
```

After running the code in the notebook, you should see the forecasted graph.



Go to your S3 service, and you should see a new file named '*forecast\_total\_water\_usage\_2024-08-17\_to\_2024-09-16.csv*' under the '*forecast*' directory in your bucket. The forecasted water usage demand is stored in this file.



The screenshot shows the Amazon S3 console interface. At the top, the navigation path is displayed as 'Amazon S3 > Buckets > water-volume-storage > forecast/'. The word 'forecast/' is circled in red. Below the path, the word 'forecast/' is shown again, followed by a 'Copy S3 URI' button. Underneath, there are two tabs: 'Objects' (which is selected and highlighted in blue) and 'Properties'. The main area is titled 'Objects (1) Info' and contains a table with one item. The table has columns for Name, Type, Last modified, Size, and Storage class. The single object listed is 'forecast\_total\_water\_usage\_2024-08-17\_to\_2024-09-16.csv', which is circled in red. The table also includes standard S3 actions like Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload. A search bar labeled 'Find objects by prefix' is located above the table, and a pagination indicator '1' is shown below it. The storage class is listed as 'Standard'.

Name	Type	Last modified	Size	Storage class
<a href="#">forecast_total_water_usage_2024-08-17_to_2024-09-16.csv</a>	csv	August 21, 2024, 10:56:19 (UTC+08:00)	1.2 KB	Standard

Here is an example of forecasted water usage data.

date	forecasted_water_volume
24/8/2024 0:00	3.145479899
25/8/2024 0:00	3.446850713
26/8/2024 0:00	3.793579094
27/8/2024 0:00	4.122465724
28/8/2024 0:00	4.458018899
29/8/2024 0:00	4.789107497
30/8/2024 0:00	5.122080414
31/8/2024 0:00	5.456221848
1/9/2024 0:00	5.78806716
2/9/2024 0:00	6.121499969
3/9/2024 0:00	6.45420208
4/9/2024 0:00	6.787247406
5/9/2024 0:00	7.120038847
6/9/2024 0:00	7.452929982
7/9/2024 0:00	7.785832858
8/9/2024 0:00	8.118653227
9/9/2024 0:00	8.451516843
10/9/2024 0:00	8.784342091
11/9/2024 0:00	9.117173006
12/9/2024 0:00	9.44998157
13/9/2024 0:00	9.782785045
14/9/2024 0:00	10.11557771
15/9/2024 0:00	10.44835761
16/9/2024 0:00	10.78112853
Total	178.3951097

# Amazon Athena

With all the data stored in S3, we can now visualize it to provide insights to stakeholders and farmers using Amazon QuickSight. Before visualizing the data, we need to preprocess it by performing SQL operations in Amazon Athena. Athena is a serverless, interactive query service that allows you to use standard SQL to query data directly from Amazon S3. Using Athena to filter or aggregate the data first helps reduce data size and complexity, resulting in faster and more efficient visualizations.

First, navigate to the S3 service and create a new bucket to store the query results. Name the bucket 'query-result-bucket' and leave the remaining settings as default.

The screenshot shows the 'Create bucket' wizard in the Amazon S3 console. The path in the top navigation bar is 'Amazon S3 > Buckets > Create bucket'. The main title is 'Create bucket' with an 'Info' link. A sub-instruction says 'Buckets are containers for data stored in S3.' Below this is a 'General configuration' section. Under 'AWS Region', it shows 'Asia Pacific (Singapore) ap-southeast-1'. The 'Bucket name' field contains 'query-result-bucket', which is highlighted with a blue border. Below the field, a note states: 'Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)' with a small icon. There is also a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button and a note about copied settings. The 'Format: s3://bucket/prefix' placeholder is shown. At the bottom, there is an 'Object Ownership' section with an 'Info' link and a note about controlling object ownership and ACLs.

Amazon S3 > Buckets > Create bucket

## Create bucket [Info](#)

Buckets are containers for data stored in S3.

### General configuration

AWS Region  
Asia Pacific (Singapore) ap-southeast-1

Bucket name [Info](#)  
query-result-bucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*  
Only the bucket settings in the following configuration are copied.

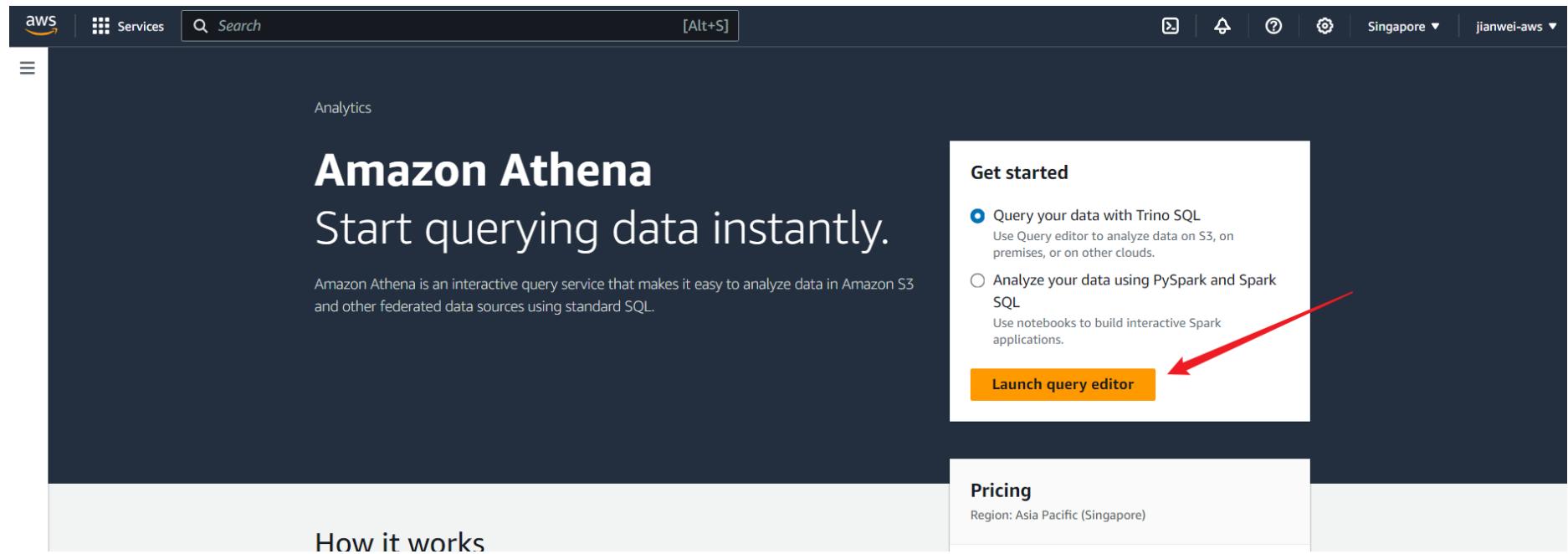
[Choose bucket](#)

Format: s3://bucket/prefix

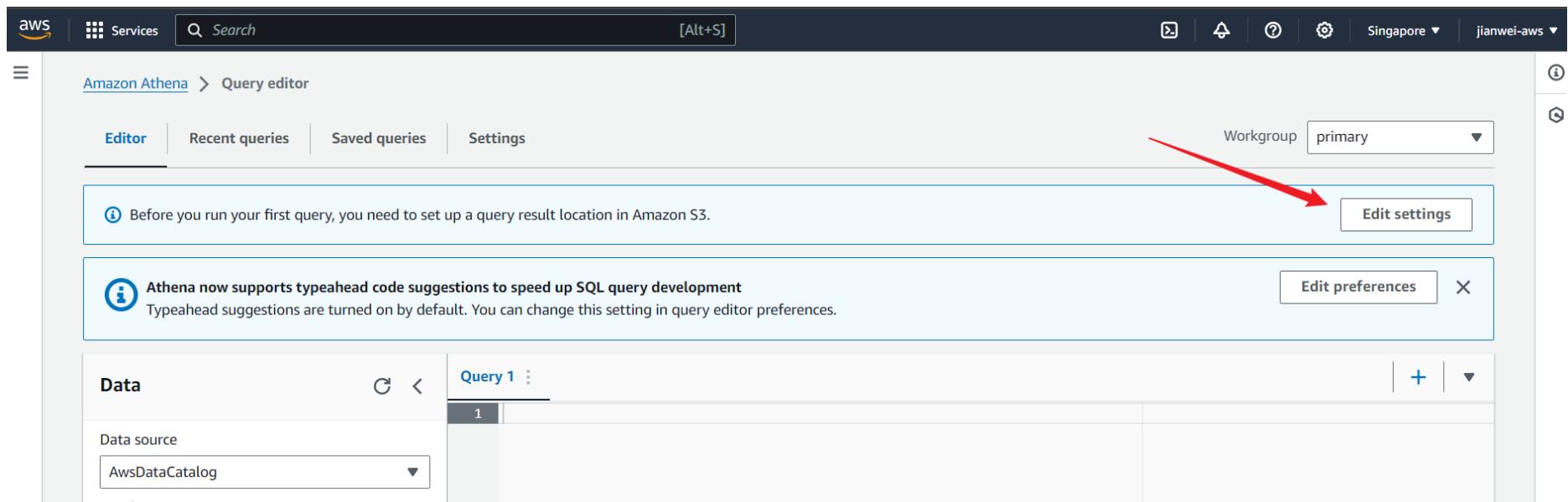
### Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

After that, type 'Athena' in the search bar and select the 'Athena' service. You will be directed to the Athena page. Choose 'Query your data in Trino SQL' since we want to analyze data in S3 and click 'Launch query editor'.



Before running your first query, you need to set up a query result location in Amazon S3. Click 'Edit settings'.



Click ‘Browse S3’ to select your bucket.

Amazon Athena > Query editor > Manage settings

## Manage settings

### Query result location and encryption

Location of query result - *optional*  
Enter an S3 prefix in the current region where the query result will be saved as an object.

View  

Expected bucket owner - *optional*  
Specify the AWS account ID that you expect to be the owner of your query results output location bucket.

Assign bucket owner full control over query results  
Enabling this option grants the owner of the S3 query results bucket full control over the query results. This means that if your query result location is owned by another account, you grant full control over your query results to the other account.

Encrypt query results

Choose the bucket that you created in previous step.

Choose S3 data set

S3 buckets

Bucket (1/3)

Name	Creation date
bucketdemounique	2024-08-23T06:39:22.000+08:00
query-result-unique-bucket	2024-08-26T12:27:23.000+08:00
water-volume-storage	2024-07-29T16:53:44.000+08:00

Cancel Choose

The screenshot shows a modal dialog titled "Choose S3 data set". Inside, there's a heading "S3 buckets" and a sub-heading "Bucket (1/3)". Below these are two columns: "Name" and "Creation date". Three rows of data are listed:

Name	Creation date
bucketdemounique	2024-08-23T06:39:22.000+08:00
query-result-unique-bucket	2024-08-26T12:27:23.000+08:00
water-volume-storage	2024-07-29T16:53:44.000+08:00

The row for "query-result-unique-bucket" is highlighted with a blue border, indicating it is selected. At the bottom right of the modal are "Cancel" and "Choose" buttons.

Click 'Save'. All query results will be saved to this bucket.

## Manage settings

### Query result location and encryption

**Location of query result - optional**  
Enter an S3 prefix in the current region where the query result will be saved as an object.

X View Browse S3

**Info** You can create and manage lifecycle rules for this bucket Lifecycle configuration

**Expected bucket owner - optional**  
Specify the AWS account ID that you expect to be the owner of your query results output location bucket.

Assign bucket owner full control over query results  
Enabling this option grants the owner of the S3 query results bucket full control over the query results. This means that if your query result location is owned by another account, you grant full control over your query results to the other account.

Encrypt query results

Cancel Save



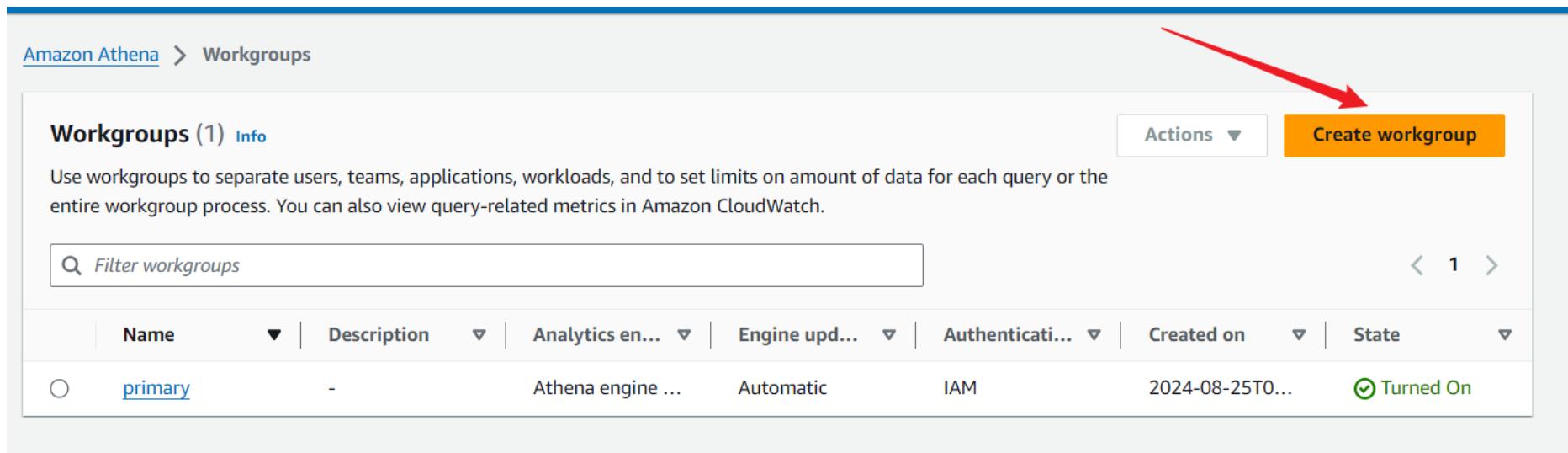
Click ‘Workgroups’ from the left navigation bar.

The screenshot shows the Amazon Athena interface. On the left, there is a navigation sidebar with the following structure:

- Amazon Athena** (with a close button)
- Query editor** (selected)
- Notebook editor
- Notebook explorer
- Jobs** (expanded)
  - Workflows
  - Powered by Step Functions
- Administration** (expanded)
  - Workgroups** (highlighted with a red arrow)
  - Capacity reservations
  - Data sources
- What's new (9+)
- Turn on compact mode

The main content area is titled "Amazon Athena > Query editor". It has tabs for Editor, Recent queries, Saved queries, and **Settings**. The **Settings** tab is active. A dropdown menu labeled "Workgroup" shows "primary". Below this, there is a section titled "Query result and encryption settings" with a "Manage" button. The settings table includes columns for Query result location (set to s3://query-result-unique-bucket/), Encrypt query results (set to "-"), Expected bucket owner (set to "-"), and Assign bucket owner full control over query results (set to "Turned off").

Click ‘Create workgroup’



The screenshot shows the 'Workgroups' page in the Amazon Athena console. At the top left, there is a breadcrumb navigation: 'Amazon Athena > Workgroups'. Below the breadcrumb, the title 'Workgroups (1) Info' is displayed. To the right of the title are two buttons: 'Actions ▾' and a prominent orange button labeled 'Create workgroup' with a red arrow pointing to it. A descriptive text block follows: 'Use workgroups to separate users, teams, applications, workloads, and to set limits on amount of data for each query or the entire workgroup process. You can also view query-related metrics in Amazon CloudWatch.' Below this text is a search bar with the placeholder 'Filter workgroups' and a pagination indicator showing '1' between arrows. The main content area contains a table with one row. The columns are: Name, Description, Analytics en..., Engine upd..., Authenticati..., Created on, and State. The row data is: primary, -, Athena engine ..., Automatic, IAM, 2024-08-25T0..., and Turned On (with a green checkmark icon).

Name	Description	Analytics en...	Engine upd...	Authenticati...	Created on	State
primary	-	Athena engine ...	Automatic	IAM	2024-08-25T0...	Turned On

Enter your workgroup name.

Amazon Athena > Workgroups > Create workgroup

## Create workgroup

**Workgroup details**

Enter a unique name for your workgroup. To change the workgroup name, delete the workgroup and recreate it with a new name.

Workgroup name

Test

Workgroup name must be from 1-128 characters and must be unique per region of your account. Valid characters are a-z, A-Z, 0-9, \_(underscore), .(period) and -(hyphen). This value cannot be changed after creation.

Description - *optional*

Workgroup description must be from 1-1024 characters. 1024 characters remaining.

**Analytics engine - *new* Info**

Choose the type of engine

Athena SQL

Apache Spark



Click 'Browse S3' to select the S3 bucket you created in the previous step. Leave the other settings as default and click 'Create workgroup'.

▼ **Query result configuration - optional**

Location of query result - *optional*  
Enter an S3 prefix in the current region where the query result will be saved as an object.

s3://bucket/prefix/object/

[View](#) 

[Browse S3](#)

Expected bucket owner - *optional*  
Specify the AWS account ID that you expect to be the owner of your query results output location bucket.

Enter AWS account ID

Assign bucket owner full control over query results  
Enabling this option grants the owner of the S3 query results bucket full control over the query results. This means that if your query result location is owned by another account, you grant full control over your query results to the other account.

Encrypt query results

Go back to 'Settings' and switch to the workgroup you just created.

The screenshot shows the Amazon Athena Query editor interface. At the top, there is a navigation bar with links for 'Amazon Athena' and 'Query editor'. Below the navigation bar, there are tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings', with 'Settings' being the active tab. A red arrow points to the 'Settings' tab. To the right of the tabs, there is a 'Workgroup' dropdown menu. The dropdown menu shows a list of workgroups: 'primary' (selected), 'Test', and 'primary'. The 'Test' workgroup is highlighted with a blue border, and a red arrow points to it. The 'primary' workgroup is also highlighted with a blue border. Below the dropdown menu, there is a section titled 'Query result and encryption settings' containing four rows of configuration options:

Query result location	Encrypt query results	Expected bucket owner	Assign bucket owner full control over query results
s3://water-volume-storage/data/	-	-	Turned off

Click the 'Editor' tab, then click 'Create' and select 'S3 bucket data'. This will manually create a table directly from your S3 bucket.

The screenshot shows the Amazon Athena Query editor interface. At the top, there is a navigation bar with 'Amazon Athena > Query editor'. Below it, a tab bar has 'Editor' selected, followed by 'Recent queries', 'Saved queries', and 'Settings'. A dropdown for 'Workgroup' is set to 'primary'. A message box at the top states: 'Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.' with 'Edit preferences' and 'X' buttons. On the left, a sidebar titled 'Data' contains sections for 'Data source' (set to 'AwsDataCatalog') and 'Database' (set to 'default'). Under 'Tables and views', there is a 'Create' button and a search bar for filtering tables and views. A dropdown menu titled 'Create a table from data source' is open, showing options: 'S3 bucket data' (which has a red arrow pointing to it), 'AWS Glue Crawler', 'Create with SQL' (with sub-options: 'CREATE TABLE', 'CREATE TABLE AS SELECT', 'CREATE TABLE AS SELECT(ICEBERG)', and 'CREATE VIEW'), and a 'Create' button. The main area is labeled 'SQL Ln 1, Col 1'.

Enter ‘water\_usage\_analysis’ as the table name.

## Create table from S3 bucket data Info

### Table details

Table name

Table name must be from 1-128 characters and must be unique. Valid characters are a-z, A-Z, 0-9, \_(underscore). Table names tend to correspond to the directory where the data will be stored.

Description - *optional*

Table description must be from 1-1024 characters. 1024 characters remaining.

You can create a new database or choose an existing database.

**Database configuration** [Info](#)

---

Choose an existing database or create a new database

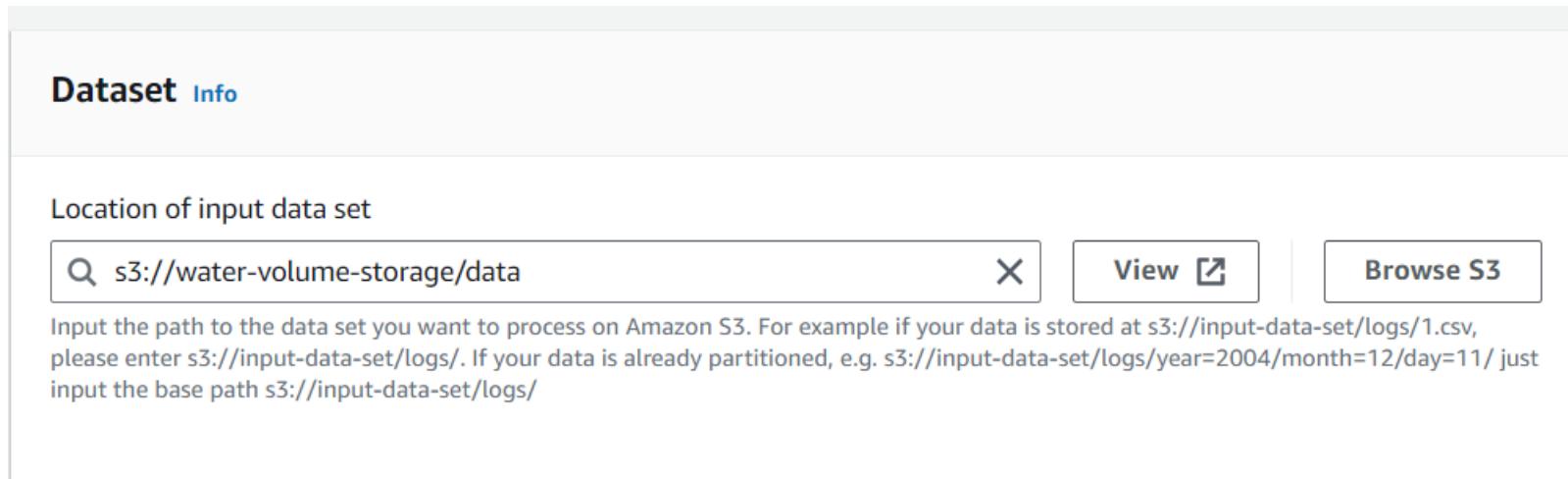
Choose to access an existing database or to create a new database in order to create a new table. Athena stores the table schema in the AWS Glue Data Catalog.

Create a database

Choose an existing database

▼

Click on ‘Browse S3’ to select the bucket where you store the data.



Select the data format as shown in the picture.

**Data format** [Info](#)

Table type

Apache Hive

File format

CSV

SerDe library

org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

SerDe properties - *optional*

Name	Value	Remove
field.delim	,	<button>Remove</button>

[Add SerDe property](#)

Enter the names and data types of all columns, then click 'Create table'.

**Column details**  
Column name must be from 1-128 characters. Valid characters are a-z, A-Z, 0-9, \_(underscore). Certain advanced column types (namely, structs) are not exposed in this interface.

Column name	Column type	Description - optional	
humidity	double	<a href="#">Enter description</a>	<a href="#">Remove</a>
temperature	double	<a href="#">Enter description</a>	<a href="#">Remove</a>
wind_speed	double	<a href="#">Enter description</a>	<a href="#">Remove</a>
soil_moisture	double	<a href="#">Enter description</a>	<a href="#">Remove</a>
location_id	string	<a href="#">Enter description</a>	<a href="#">Remove</a>
date_time	string	<a href="#">Enter description</a>	<a href="#">Remove</a>
water_volume	double	<a href="#">Enter description</a>	<a href="#">Remove</a>

[Add a column](#) [Bulk add columns](#)

Alternatively, you can run the following SQL statement to create a table.

```
CREATE EXTERNAL TABLE IF NOT EXISTS `water-usage`.`water_usage_analysis` (
    `humidity` DOUBLE,
    `temperature` DOUBLE,
    `wind_speed` DOUBLE,
    `soil_moisture` DOUBLE,
    `location_id` STRING,
    `date_time` STRING,
    `water_volume` DOUBLE
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
    'separatorChar' = ',',
    'quoteChar' = '\"'
)
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://bucketdemounique/input/'

TBLPROPERTIES ('classification' = 'csv');
```

You will see that a new ‘water\_usage\_analysis’ table has been successfully created.

The screenshot shows the AWS Athena Query Editor interface. The top navigation bar includes tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings', along with a 'Workgroup' dropdown set to 'Test'. A message通知 states: 'Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.' Below the message, the main workspace displays a 'Data' sidebar on the left with sections for 'Data source' (set to 'AwsDataCatalog') and 'Database' (set to 'water-usage'). The central area contains a code editor with the following SQL script:

```
1 ✓ CREATE EXTERNAL TABLE IF NOT EXISTS `water-usage`.`water_usage_analysis` (
2     `humidity` DOUBLE,
3     `temperature` DOUBLE,
4     `wind_speed` DOUBLE,
5     `soil_moisture` DOUBLE,
6     `location_id` STRING,
7     `date_time` STRING,
8     `water_volume` DOUBLE
9 )
10 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
11 WITH SERDEPROPERTIES (
12     'separatorChar' = ',',
13     'quotechar' = '\"'
14 )
15 STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
```

A red arrow points to the status bar at the bottom of the code editor, which displays 'SQL Ln 8, Col 24'. Below the code editor are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. To the right of the code editor, there are icons for 'Reuse query results' (with a note 'up to 60 minutes ago'), 'Copy', 'Edit', and 'Delete'. At the very bottom of the editor, tabs for 'Query results' and 'Query stats' are visible.

Now, we can perform an SQL query on the table to filter the data we want to visualize in QuickSight. Run the following SQL statement:

```
SELECT
CASE
    WHEN location_id = 'region_1' THEN 'Johor'
    WHEN location_id = 'region_2' THEN 'Perak'
    WHEN location_id = 'region_3' THEN 'Selangor'
    WHEN location_id = 'region_4' THEN 'Kuantan'
    WHEN location_id = 'region_5' THEN 'Sabah'
    ELSE location_id
END AS location_id,
DATE_FORMAT(
    parse_datetime(date_time, 'd/M/yyyy HH:mm'),
    '%Y-%m'
) AS month,
SUM(water_volume) AS total_water_usage
FROM
"water-usage"."water_usage_analysis"
GROUP BY
location_id,
DATE_FORMAT(
    parse_datetime(date_time, 'd/M/yyyy HH:mm'),
    '%Y-%m'
)
ORDER BY
location_id,
DATE_FORMAT(
    parse_datetime(date_time, 'd/M/yyyy HH:mm'),
    '%Y-%m'
)\.
```

## The query result:

Query resultsQuery stats

⌚ CompletedTime in queue: 65 msRun time: 644 msData scanned: 35.35 KB

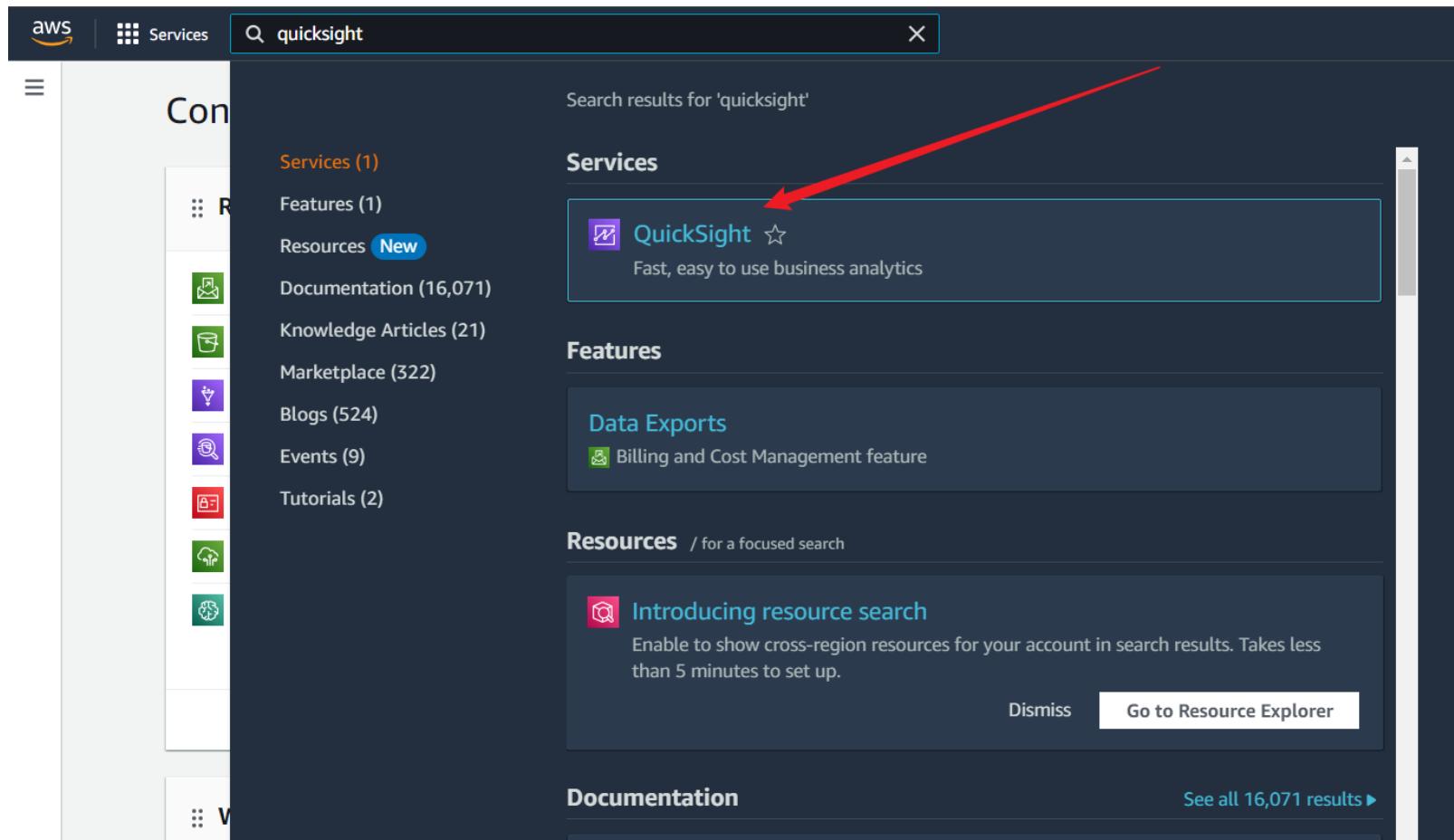
Results (25)CopyDownload results

Search rows< 1 >⚙️

#	location_id	month	total_water_usage
1	Johor	2024-04	740.210000000002
2	Johor	2024-05	762.05
3	Johor	2024-06	712.2
4	Johor	2024-07	736.48
5	Johor	2024-08	773.369999999998
6	Kuantan	2024-04	731.750000000001
7	Kuantan	2024-05	730.32
8	Kuantan	2024-06	769.060000000001
9	Kuantan	2024-07	774.800000000002
10	Kuantan	2024-08	754.890000000002
11	Perak	2024-04	759.719999999999

# Amazon QuickSight

Type ‘QuickSight’ in search bar and click the ‘QuickSight’ service.



Click ‘Sign up for QuickSight’

Your AWS Account is not signed up for QuickSight. Would you like to sign up now?

AWS Account

Sign up for QuickSight



To access QuickSight with a different account, [log in](#) again.

Choose the ‘Enterprise’ edition and press ‘Continue’.

### Create your QuickSight account



#### Enterprise edition offers the Paginated Reports add-on

When you sign up for the Enterprise edition, you have the option to add Paginated Reports to your subscription. Reporting enables your business users to generate paginated documents to be sent to recipients in a number of formats.

[Learn more](#)

#### Edition

#### Enterprise

#### Enterprise + Q

[Learn more](#)

Team trial for 30 days (4 authors)\*

**FREE**

**FREE**

Author per month (yearly)\*\*

\$18

\$28

Author per month (monthly)\*\*

\$24

\$34

Readers (pay-per-Session)

\$0.30 / session (max \$5)\*\*\*\*

\$0.30 / session (max \$10)\*\*\*\*

Additional SPICE per month

\$0.38 per GB

\$0.38 per GB

QuickSight Q regional fee

N/A

\$250 / mo / region

1. Select ‘Singapore’ for the region.
2. Enter your account name.
3. Enter your notification email.

### Create your QuickSight account

Enterprise

[Back](#)

#### Authentication method

- Use IAM federated identities & QuickSight-managed users  
Authenticate with single sign-on (SAML or OpenID Connect), AWS IAM credentials, or QuickSight credentials
- Use IAM federated identities only  
Authenticate with single sign-on (SAML or OpenID Connect) or AWS IAM credentials
- Use Active Directory  
Authenticate with Active Directory credentials

#### QuickSight region

Select a region

1

#### Account info

QuickSight account name

You will need this for you and others to sign in

2

Notification email address

For QuickSight to send important notifications

3

#### QuickSight access to AWS services

Make your existing AWS data and users available in QuickSight. [Learn more](#)

Click on 'Select S3 buckets' and choose the S3 bucket that contains the water usage data. Leave the other settings as default and click 'Finish'.

#### QuickSight access to AWS services

Make your existing AWS data and users available in QuickSight. [Learn more](#)

##### IAM Role

- Use QuickSight-managed role (default)  
 Use an existing role

##### Allow access and autodiscovery for these resources

-  Amazon Redshift  
  Amazon RDS  
  IAM  
  Amazon S3 (1 buckets selected)  
[Select S3 buckets](#)  
  Amazon Athena  
Make sure you've chosen the right Amazon S3 buckets for QuickSight access  
  Amazon S3 Storage Analytics  
  AWS IoT Analytics  
  Amazon OpenSearch Service  
  Amazon SageMaker  
  Amazon Timestream  
  AWS SecretsManager  
[Select secrets](#)

Finish

You have now successfully created a QuickSight account.

Congratulations! You are signed up for Amazon QuickSight!

Access QuickSight with the following information

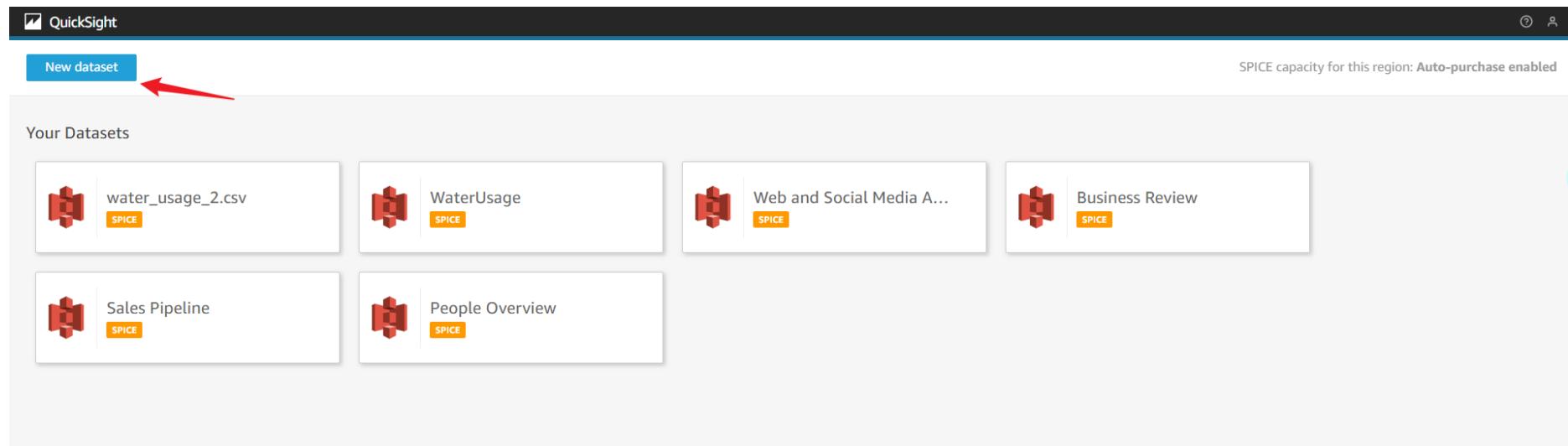
Account name: techwithlucy

[Go to Amazon QuickSight](#)

Click ‘New analysis’.

The screenshot shows the Amazon QuickSight interface. On the left, there is a sidebar with navigation links: 'Favorites', 'Recent', 'My folders', 'Shared folders', 'Dashboards', 'Analyses' (which is selected and highlighted in blue), and 'Datasets'. The main area is titled 'Analyses' and displays six analysis cards. Each card includes a thumbnail, the analysis name, and an 'Updated' timestamp. The first analysis is 'water\_usage\_2.csv analysis' (updated 16 days ago). The second is 'WaterUsage analysis'. The third is 'Business Review analysis' (labeled as a 'SAMPLE'). The fourth is 'People Overview analysis' (labeled as a 'SAMPLE'). The fifth is 'Sales Pipeline analysis' (labeled as a 'SAMPLE'). The sixth is 'Web and Social Media Anal...' (labeled as a 'SAMPLE'). At the top right of the main area, there is a 'Last updated (newest first)' dropdown, a grid icon, a list icon, and a 'New analysis' button, which is highlighted with a red arrow pointing to it.

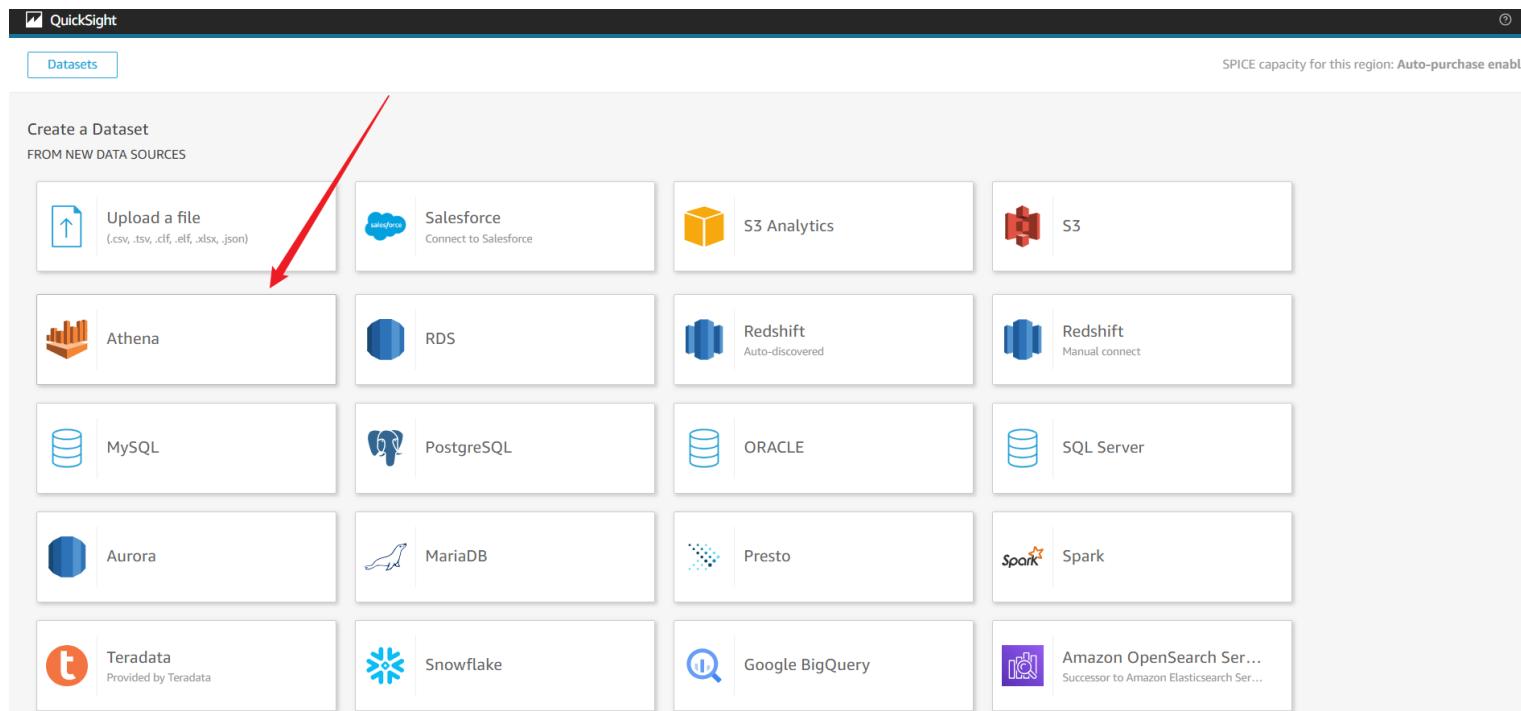
Select ‘New dataset’.



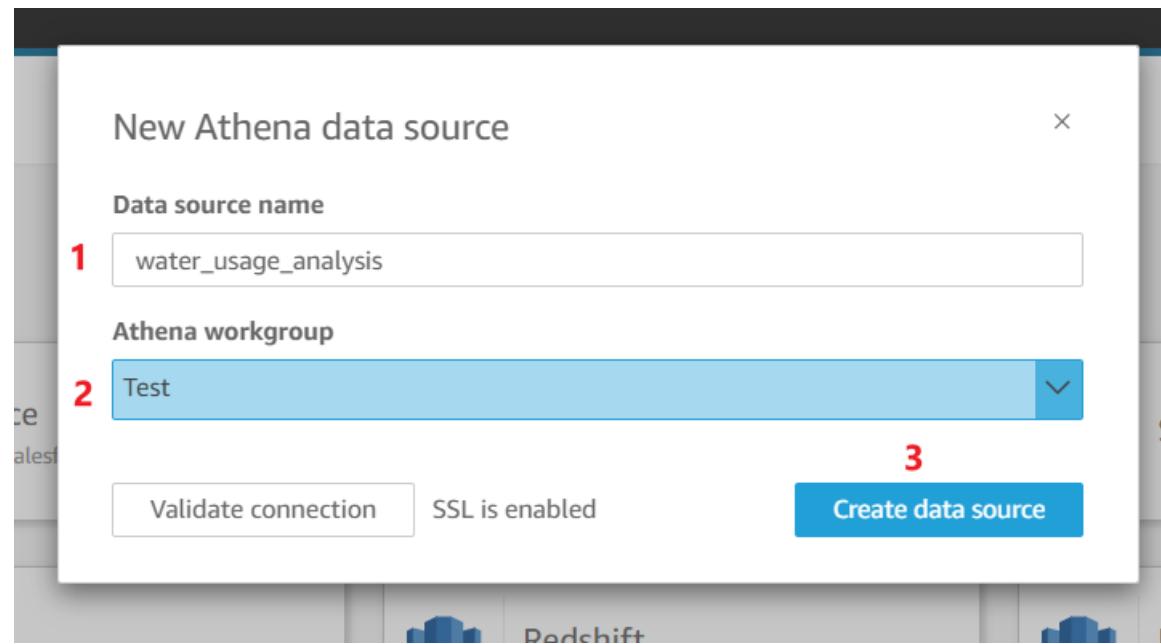
The screenshot shows the Amazon QuickSight console interface. At the top, there's a dark header bar with the 'Quicksight' logo and some icons. Below it, the main content area has a light gray background. In the top-left of this area, there's a blue rectangular button with white text that says 'New dataset'. A red arrow points from the left towards this button. To the right of the button, the text 'SPICE capacity for this region: Auto-purchase enabled' is displayed. Below the button, the heading 'Your Datasets' is centered. Underneath this heading, there are six dataset cards arranged in two rows of three. Each card features a small red cube icon on the left, followed by the dataset name and a small orange 'SPICE' badge on the right. The datasets listed are: 'water\_usage\_2.csv', 'WaterUsage', 'Web and Social Media A...', 'Business Review', 'Sales Pipeline', and 'People Overview'.

Your Datasets		
water_usage_2.csv SPICE	WaterUsage SPICE	Web and Social Media A... SPICE
Sales Pipeline SPICE	People Overview SPICE	Business Review SPICE

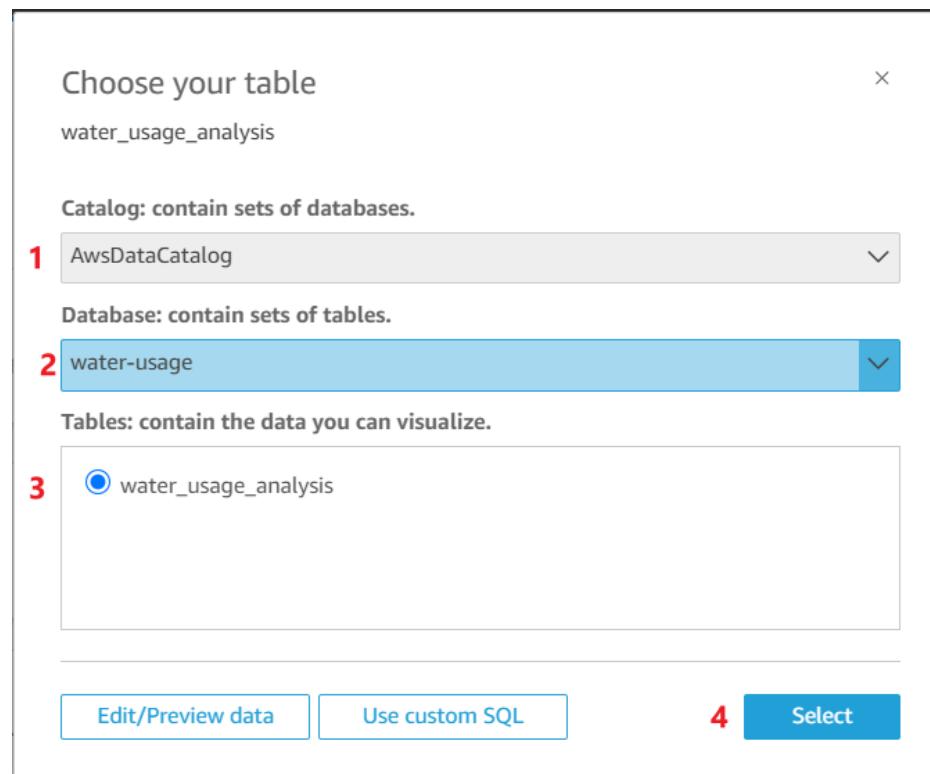
Select ‘Anthena’.



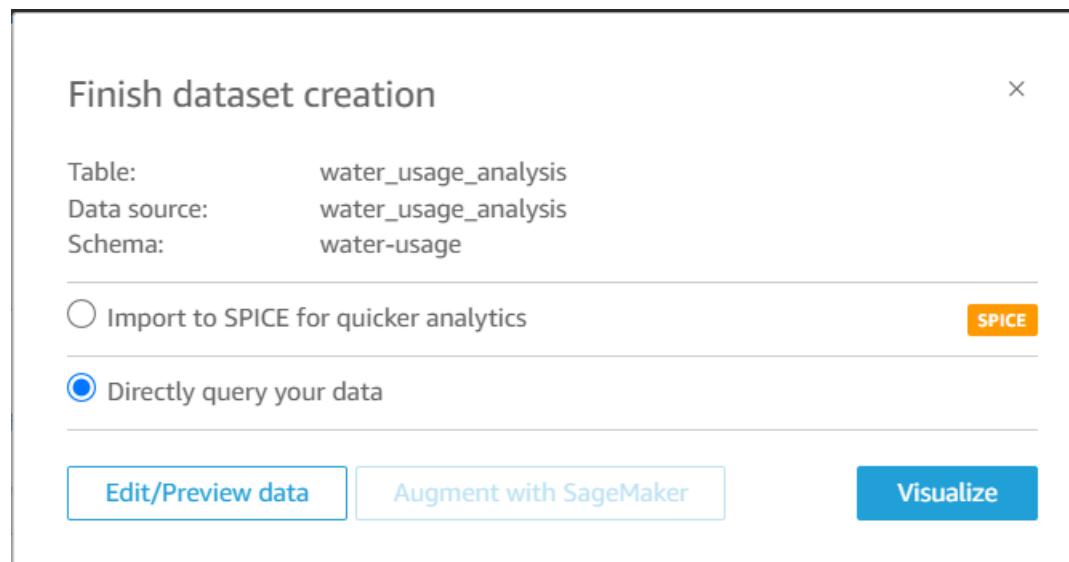
1. Enter ‘water\_usage\_analysis’ as data source name.
2. Select ‘Test’ as Athena workgroup.
3. Click ‘Create data source’.



1. Select ‘AwsDataCatalog’
2. Select ‘water-usage’ database.
3. Select ‘water\_usage\_analysis’ table.



Choose ‘Directly query your data’ and click ‘Visualize’.



1. You can choose multiple data that want to be visualized.
2. You can choose various types of visualization such as bar chart, histogram, line chart, etc.

The screenshot shows the Amazon QuickSight interface with the following components:

- Data Panel (Left):** Shows the "water\_usage\_summary" dataset selected. Under "ROWS", "month" is selected. Under "VALUES", "# total\_water\_usage" is selected and highlighted with an orange border.
- Visuals Panel (Middle):** Shows the "Pivot table" selected under "CHANGE VISUAL TYPE". The "ROWS" section has "month" assigned. The "VALUES" section has "# total\_water\_usage" assigned.
- Sheet 1 (Right):** Displays a table titled "Sum of Total\_water\_usage by Month". The table has one row with the value 3,796.98.
- Visual Selection Panel (Bottom):** A grid of icons representing different visualization types, with a tooltip "Table: requires 1 dimension in Group by or 1 measure in" pointing to the table icon.

After you finish customizing your visualization, you can click the 'PUBLISH' button.

QuickSight | water\_usage\_summary analysis

File Edit Data Insert Sheets Objects Search

ACTUAL SIZE ▾ PUBLISH NEW LOOK ▾

Sheet 1 +

Sum of Total\_water\_usage by Month

Rows	total_water_usage
2024-08	3,796.98
2024-07	3,723.46
2024-06	3,723.91
2024-05	3,730.57
2024-04	3,724.58

Sum of Total\_water\_usage by Location\_id

A pie chart titled "Sum of Total\_water\_usage by Location\_id". The chart is divided into five segments, each representing a different location. The segments are colored and labeled: Selangor (light blue), Kuantan (dark blue), Johor (light green), Perak (orange), and Sabah (pink). The chart is grouped by location\_id and sized by the sum of total\_water\_usage. A legend on the right side lists the locations with their corresponding colors: Selangor (light blue), Kuantan (dark blue), Johor (light green), Perak (orange), and Sabah (pink).

Group By: location\_id  
Size: total\_water\_usage (Sum)

Here is the completed dashboard.

