

# Building an IoT Data-to-Dashboard Solution with AWS IoT Core, Timestream, and Grafana

## Table of Contents

1. Introduction
  2. Prerequisites
  3. Step-by-Step Implementation
    1. Creating an AWS IoT Thing
    2. Connecting the IoT Device and Sending Telemetry Data
    3. Storing Data in Amazon Timestream
    4. Visualizing Data with Amazon Managed Grafana
  4. Summary
  5. Additional Resources
- 

## 1. Introduction

This document outlines a step-by-step guide on creating an IoT system using AWS services to collect, store, and visualize real-time data. By leveraging AWS IoT Core, Amazon Timestream, and Amazon Managed Grafana, we will demonstrate how to create an end-to-end IoT data solution that is both scalable and secure.

## High-Level Overview

1. Create an **AWS IoT Core Thing** (a virtual representation of your IoT device).
2. **Send telemetry data** from the IoT device (laptop) to AWS IoT Core using the AWS IoT Python SDK.
3. **Store telemetry data in Amazon Timestream**.
4. **Visualize the data in Amazon Managed Grafana** using time-series dashboards.

## 2. Prerequisites

To follow along with this guide, you will need:

- **Device with Python installed:** Your laptop will act as the IoT device.
  - **AWS Account:** Make sure you have an active AWS account with permissions to use AWS IoT Core, Amazon Timestream, and Amazon Managed Grafana.
  - **Basic understanding of AWS services and Python.**
- 5.

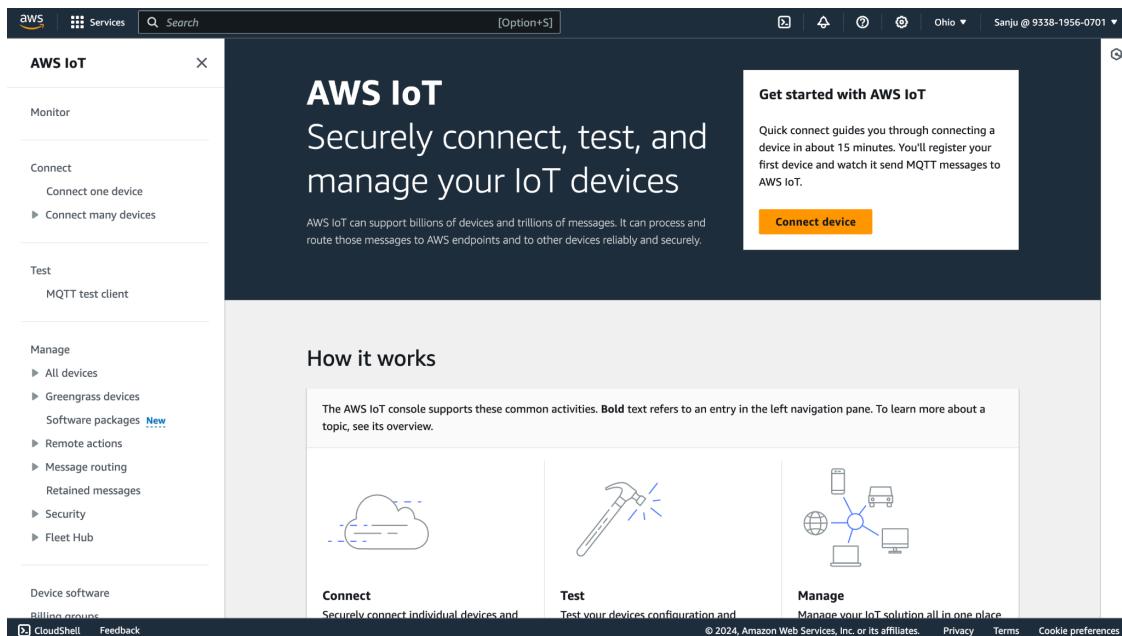
### 3. Step-by-Step Implementation

#### Open AWS Console:

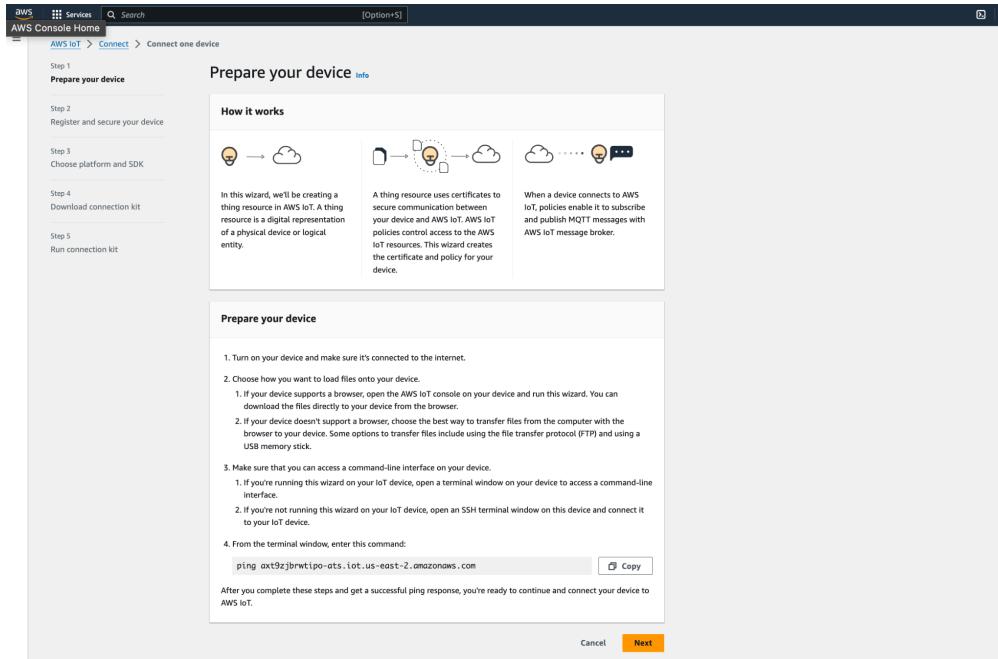
- Go to the AWS Console and sign in with your AWS credentials.
- Search for AWS IoT Core and open it.

#### Creating an AWS IoT Thing

- Log in to the AWS and search for IoT Core console.

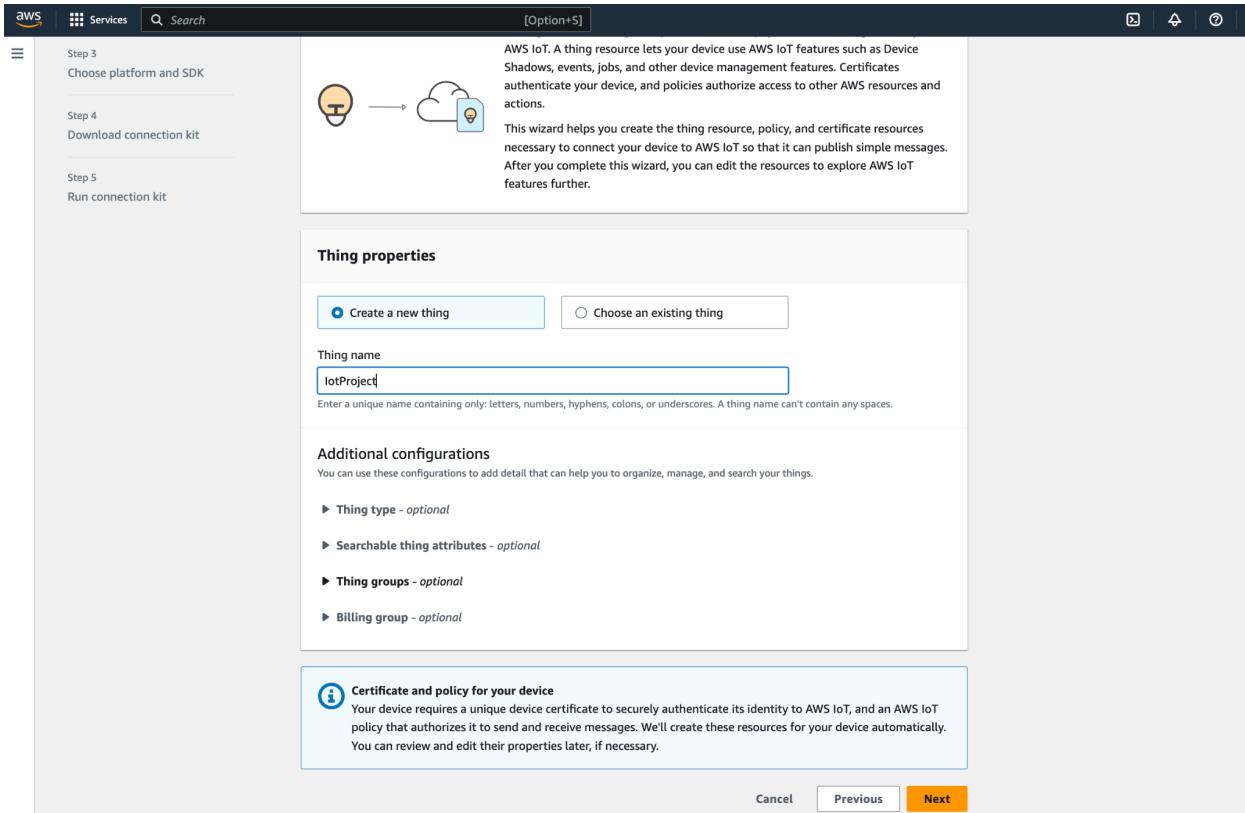


- Click on Connect Device and Please Copy the code and enter in the terminal on the laptop.



```
(base) saisrivatsat@Sanjus-Air AWS Projects % ping axt9zjbrwtipo-ats.iot.us-east-2.amazonaws.com
PING axt9zjbrwtipo-ats.iot.us-east-2.amazonaws.com (18.223.79.136): 56 data bytes
64 bytes from 18.223.79.136: icmp_seq=0 ttl=244 time=43.569 ms
64 bytes from 18.223.79.136: icmp_seq=1 ttl=244 time=26.457 ms
64 bytes from 18.223.79.136: icmp_seq=2 ttl=244 time=25.901 ms
64 bytes from 18.223.79.136: icmp_seq=3 ttl=244 time=26.686 ms
64 bytes from 18.223.79.136: icmp_seq=4 ttl=244 time=31.714 ms
64 bytes from 18.223.79.136: icmp_seq=5 ttl=244 time=26.924 ms
64 bytes from 18.223.79.136: icmp_seq=6 ttl=244 time=28.905 ms
64 bytes from 18.223.79.136: icmp_seq=7 ttl=244 time=27.019 ms
64 bytes from 18.223.79.136: icmp_seq=8 ttl=244 time=26.661 ms
64 bytes from 18.223.79.136: icmp_seq=9 ttl=244 time=34.454 ms
64 bytes from 18.223.79.136: icmp_seq=10 ttl=244 time=26.004 ms
64 bytes from 18.223.79.136: icmp_seq=11 ttl=244 time=26.620 ms
64 bytes from 18.223.79.136: icmp_seq=12 ttl=244 time=28.043 ms
64 bytes from 18.223.79.136: icmp_seq=13 ttl=244 time=27.508 ms
64 bytes from 18.223.79.136: icmp_seq=14 ttl=244 time=28.143 ms
64 bytes from 18.223.79.136: icmp_seq=15 ttl=244 time=32.476 ms
64 bytes from 18.223.79.136: icmp_seq=16 ttl=244 time=32.136 ms
64 bytes from 18.223.79.136: icmp_seq=17 ttl=244 time=28.181 ms
64 bytes from 18.223.79.136: icmp_seq=18 ttl=244 time=32.700 ms
64 bytes from 18.223.79.136: icmp_seq=19 ttl=244 time=31.888 ms
64 bytes from 18.223.79.136: icmp_seq=20 ttl=244 time=26.860 ms
64 bytes from 18.223.79.136: icmp_seq=21 ttl=244 time=32.729 ms
64 bytes from 18.223.79.136: icmp_seq=22 ttl=244 time=26.363 ms
64 bytes from 18.223.79.136: icmp_seq=23 ttl=244 time=27.290 ms
64 bytes from 18.223.79.136: icmp_seq=24 ttl=244 time=27.664 ms
64 bytes from 18.223.79.136: icmp_seq=25 ttl=244 time=29.083 ms
64 bytes from 18.223.79.136: icmp_seq=26 ttl=244 time=27.419 ms
64 bytes from 18.223.79.136: icmp_seq=27 ttl=244 time=33.760 ms
64 bytes from 18.223.79.136: icmp_seq=28 ttl=244 time=32.175 ms
64 bytes from 18.223.79.136: icmp_seq=29 ttl=244 time=32.794 ms
```

- Select Create a new thing naming as IoTProject and click next



- Please select platform and SDK, AWS IoT Core will generate a connection kit that includes all necessary certificates, keys, and a sample Python SDK application.

AWS IoT > Connect > Connect one device

Step 1  
[Prepare your device](#)

Step 2  
[Register and secure your device](#)

Step 3  
**Choose platform and SDK**

Step 4  
Download connection kit

Step 5  
Run connection kit

## Choose platform and SDK [Info](#)

### Choose the software for your device



This wizard helps you download a software development kit (SDK) to your device. AWS IoT supports Device SDKs that run on your device and include a sample program that publishes and subscribes to MQTT messages. AWS IoT supports Device SDKs in the languages shown below.

### Platform and SDK

Choose the platform OS and AWS IoT Device SDK that you want to use for your device.

**Device platform operating system**  
 This is the operating system installed on the device that will connect to AWS.

**Linux / macOS**  
 Linux version: any  
 macOS version: 10.13+

**Windows**  
 Version 10

**AWS IoT Device SDK**  
 Choose a Device SDK that's in a language your device supports.

**Node.js**  
 Version 10+  
 Requires Node.js and npm to be installed

**Python**  
 Version 3.6+  
 Requires Python and Git to be installed

**Java**  
 Version 8  
 Requires Java JDK, Maven, and Git to be installed

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

- Download the connection kit, including the **x509 certificate, public/private keys, and policy**.

The screenshot shows the AWS IoT Device SDK connection kit download page. The top navigation bar includes 'aws', 'Services', 'Search', and 'Option+S'. Below the navigation, a sidebar lists steps: Step 2 ('Register and secure your device'), Step 3 ('Choose platform and SDK'), Step 4 ('Download connection kit'), and Step 5 ('Run connection kit'). The main content area is titled 'Install the software on your device' and contains a diagram showing a ZIP file icon pointing to a device icon. A text box explains that a connection kit is created and includes a zipped file for installation. The 'Connection kit' section details the contents: Certificate (IoTProject.cert.pem), Private key (IoTProject.private.key), AWS IoT Device SDK (Python), Script to send and receive messages (start.sh), and Policy (IoTProject-Policy). A 'View policy' link is also present. A 'Download' section provides instructions for browser users and a 'Download connection kit' button. The 'Unzip connection kit on your device' section contains a ZIP file icon, a command line input field with 'unzip connect\_device\_package.zip', a 'Copy' button, and a 'Cancel' button. At the bottom right are 'Previous' and 'Next' buttons, with 'Next' highlighted.

- Skip optional configurations and download the connection kit generated by AWS IoT Core.
- Unzip the connection kit on your laptop with the code given.

```

● ○ ● AWS Projects -- -zsh -- 117x32
64 bytes from 18.223.79.136: icmp_seq=296 ttl=244 time=35.873 ms
64 bytes from 18.223.79.136: icmp_seq=297 ttl=244 time=28.377 ms
64 bytes from 18.223.79.136: icmp_seq=298 ttl=244 time=27.368 ms
64 bytes from 18.223.79.136: icmp_seq=299 ttl=244 time=34.412 ms
64 bytes from 18.223.79.136: icmp_seq=300 ttl=244 time=33.172 ms
64 bytes from 18.223.79.136: icmp_seq=301 ttl=244 time=27.133 ms
64 bytes from 18.223.79.136: icmp_seq=302 ttl=244 time=35.938 ms
64 bytes from 18.223.79.136: icmp_seq=303 ttl=244 time=25.844 ms
64 bytes from 18.223.79.136: icmp_seq=304 ttl=244 time=27.386 ms
64 bytes from 18.223.79.136: icmp_seq=305 ttl=244 time=34.795 ms
64 bytes from 18.223.79.136: icmp_seq=306 ttl=244 time=26.782 ms
64 bytes from 18.223.79.136: icmp_seq=307 ttl=244 time=32.593 ms
64 bytes from 18.223.79.136: icmp_seq=308 ttl=244 time=26.396 ms
64 bytes from 18.223.79.136: icmp_seq=309 ttl=244 time=26.566 ms
64 bytes from 18.223.79.136: icmp_seq=310 ttl=244 time=27.596 ms
64 bytes from 18.223.79.136: icmp_seq=311 ttl=244 time=27.513 ms
64 bytes from 18.223.79.136: icmp_seq=312 ttl=244 time=27.276 ms
64 bytes from 18.223.79.136: icmp_seq=313 ttl=244 time=26.892 ms
64 bytes from 18.223.79.136: icmp_seq=314 ttl=244 time=27.902 ms
64 bytes from 18.223.79.136: icmp_seq=315 ttl=244 time=26.979 ms
^C
--- axt9zjbrwrtipo-ats.iot.us-east-2.amazonaws.com ping statistics ---
316 packets transmitted, 316 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 25.049/30.166/57.330/3.732 ms
(base) saisrivatsat@Sanjus-Air AWS Projects % unzip connect_device_package.zip
Archive: connect_device_package.zip
extracting: IotProject.cert.pem
extracting: IotProject.public.key
extracting: IotProject.private.key
extracting: IotProject-Policy
extracting: start.sh
(base) saisrivatsat@Sanjus-Air AWS Projects %

```

- Run the provided script in your terminal that will allow us to display messages from device.

The screenshot shows the AWS IoT Connect one device page. The top navigation bar includes 'Services' and a search bar. A green banner at the top states: 'AWS IoT successfully created thing resource IotProject and generated your connection kit.' Below this, the main content area has a title 'Run connection kit' with an 'Info' link. On the left, a sidebar lists steps: Step 1 'Prepare your device', Step 2 'Register and secure your device', Step 3 'Choose platform and SDK', Step 4 'Download connection kit', and Step 5 'Run connection kit'. The main content area contains several sections: 'How to display messages from your device', 'Step 1: Add execution permissions', 'Step 2: Run the start script', and 'Step 3: Return to this screen to view your device's messages'. It also features a terminal window showing command inputs like 'chmod +x start.sh' and './start.sh', along with a 'Copy' button for each. At the bottom, there's a 'Subscriptions' section with 'sdk/test/python' listed, a 'Pause' button, a 'Clear' button, and a status message 'Waiting for messages'.



AWS Projects — python3 < start.sh — 117x32

```
(base) saisrivatsat@Sanjus-Air AWS Projects % chmod +x start.sh
(base) saisrivatsat@Sanjus-Air AWS Projects % ./start.sh

Downloading AWS IoT Root CA certificate from AWS...
  % Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total Spent   Left  Speed
100  1188  100  1188    0     0  9102      0 --:--:-- --:--:-- --:--:--  9138

Cloning the AWS SDK...
Cloning into 'aws-iot-device-sdk-python-v2'...
remote: Enumerating objects: 2666, done.
remote: Counting objects: 100% (1137/1137), done.
remote: Compressing objects: 100% (368/368), done.
remote: Total 2666 (delta 887), reused 882 (delta 746), pack-reused 1529 (from 1)
Receiving objects: 100% (2666/2666), 2.24 MiB | 14.10 MiB/s, done.
Resolving deltas: 100% (1730/1730), done.

Installing AWS SDK...
Processing ./aws-iot-device-sdk-python-v2
  Preparing metadata (setup.py) ... done
Collecting awscrt==0.21.1 (from awsiotsdk==1.0.0.dev0)
  Downloading awscrt-0.21.1-cp311-abi3-macosx_10_9_universal2.whl.metadata (3.4 kB)
  Downloading awscrt-0.21.1-cp311-abi3-macosx_10_9_universal2.whl (1.5 MB)
                                             1.5/1.5 MB 14.0 MB/s eta 0:00:00
Building wheels for collected packages: awsiotsdk
  Building wheel for awsiotsdk (setup.py) ... done
    Created wheel for awsiotsdk: filename=awsiotsdk-1.0.0.dev0-py3-none-any.whl size=75037 sha256=3cb87e97d76358b9acaca6cb328e6f4220168cac145204708ca227544
      Stored in directory: /Users/saisrivatsat/Library/Caches/pip/wheels/7a/0a/5c/870aa1e517e577576c205efcb14e52fc554d0f48d16
Successfully built awsiotsdk
```



AWS Projects — python3 < start.sh — 117x32

```
Sending messages until program killed
Publishing message to topic 'sdk/test/python': Hello World! [1]
Received message from topic 'sdk/test/python': b'Hello World! [1]'''
Publishing message to topic 'sdk/test/python': Hello World! [2]
Received message from topic 'sdk/test/python': b'Hello World! [2]'''
Publishing message to topic 'sdk/test/python': Hello World! [3]
Received message from topic 'sdk/test/python': b'Hello World! [3]'''
Publishing message to topic 'sdk/test/python': Hello World! [4]
Received message from topic 'sdk/test/python': b'Hello World! [4]'''
Publishing message to topic 'sdk/test/python': Hello World! [5]
Received message from topic 'sdk/test/python': b'Hello World! [5]'''
Publishing message to topic 'sdk/test/python': Hello World! [6]
Received message from topic 'sdk/test/python': b'Hello World! [6]'''
Publishing message to topic 'sdk/test/python': Hello World! [7]
Received message from topic 'sdk/test/python': b'Hello World! [7]'''
Publishing message to topic 'sdk/test/python': Hello World! [8]
Received message from topic 'sdk/test/python': b'Hello World! [8]'''
Publishing message to topic 'sdk/test/python': Hello World! [9]
Received message from topic 'sdk/test/python': b'Hello World! [9]'''
Publishing message to topic 'sdk/test/python': Hello World! [10]
Received message from topic 'sdk/test/python': b'Hello World! [10]'''
Publishing message to topic 'sdk/test/python': Hello World! [11]
Received message from topic 'sdk/test/python': b'Hello World! [11]'''
Publishing message to topic 'sdk/test/python': Hello World! [12]
Received message from topic 'sdk/test/python': b'Hello World! [12]'''
Publishing message to topic 'sdk/test/python': Hello World! [13]
Received message from topic 'sdk/test/python': b'Hello World! [13]'''
Publishing message to topic 'sdk/test/python': Hello World! [14]
Received message from topic 'sdk/test/python': b'Hello World! [14]'''
Publishing message to topic 'sdk/test/python': Hello World! [15]
Received message from topic 'sdk/test/python': b'Hello World! [15]'''
```

- Click on MQTT test client on AWS IoT console and put sdk/test/python in the topic filter then click subscribe.

The screenshot shows the AWS IoT MQTT test client interface. A green banner at the top indicates "AWS IoT successfully created thing resource IoTProject and generated your connection kit." The left sidebar has sections for Monitor, Connect (with options for one device or many devices), and Test (MQTT test client). The main content area shows the "MQTT test client" page with a "Connection details" section showing a green "Connected" status. Below it is a "Subscribe to a topic" section with a topic filter "sdk/test/python" and a "Subscribe" button. Further down is a "Subscriptions" section for "sdk/test/python" with a message payload example: 
 

```
{
      "message": "Hello from AWS IoT console"
    }
```

- Next In the IoT Core dashboard, go to **Manage > Things > IoTProject > Certificates**

The screenshot shows the AWS IoT Things page for the "iotProject" thing. The left sidebar includes sections for Monitor, Connect, and Test (MQTT test client). Under "Manage", there are sections for All devices (Things), Thing groups, Thing types, Fleet metrics, Greengrass devices, Software packages, Remote actions, Message routing, Retained messages, Security, and Fleet Hub. The main content area shows the "iotProject" thing details with fields for Name (iotProject), ARN (arn:aws:iot:us-east-2:933819560701:thing/iotProject), Type (-), and Billing group (-). Below this is a "Certificates" tab showing one certificate attached: 
 

Certificate ID	Status
Oaab0d4e27fbe50701a16576a25842b2a873bdf1a4b9932658080c0102c49ab	Active

AWS IoT successfully created thing resource **iotProject** and generated your connection kit.

[AWS IoT](#) > [Security](#) > [Certificates](#) > **Oaaab0d4e27fbe50701a16576a25842b2aa73bdf1a4b9932658080c0102c49ab** [Info](#)

**Details**

Certificate ID <b>Oaaab0d4e27fbe50701a16576a25842b2aa73bdf1a4b9932658080c0102c49ab</b>	Status <b>Active</b>
Certificate ARN <b>arn:aws:iot:us-east-2:933819560701:cert/Oaaab0d4e27fbe50701a16576a25842b2aa73bdf1a4b9932658080c0102c49ab</b>	Created <b>September 08, 2024, 02:20:35 (UTC-05:00)</b>
Subject <b>CN=AWS IoT Certificate</b>	Valid <b>September 08, 2024, 02:18:35 (UTC-05:00)</b>
Issuer <b>OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US</b>	Expires <b>December 31, 2049, 17:59:59 (UTC-06:00)</b>

**Policies** [\(1\) Info](#)

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

<input type="checkbox"/> Name
<input checked="" type="checkbox"/> <b>iotProject-Policy</b>

[Edit active version](#) [Delete](#)

AWS IoT successfully created thing resource **iotProject** and generated your connection kit.

[AWS IoT](#) > [Security](#) > [Policies](#) > **iotProject-Policy**

**iotProject-Policy** [Info](#)

**Details**

Policy ARN <b>arn:aws:iot:us-east-2:933819560701:policy/iotProject-Policy</b>	Active version <b>1</b>	Created <b>September 08, 2024, 02:20:35 (UTC-05:00)</b>	Last updated <b>September 08, 2024, 02:20:35 (UTC-05:00)</b>
--	----------------------------	--	---

**Versions** [Targets](#) [Noncompliance](#) [Tags](#)

**Active version: 1**

Policy effect	Policy action	Policy resource
Allow	iot:Publish	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/java
Allow	iot:Publish	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/python
Allow	iot:Publish	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/js
Allow	iot:Receive	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/java
Allow	iot:Receive	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/python
Allow	iot:Receive	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/js
Allow	iot:PublishRetain	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/java
Allow	iot:PublishRetain	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/python
Allow	iot:PublishRetain	arn:aws:iot:us-east-2:933819560701:topic/sdk/test/js
Allow	iot:Subscribe	arn:aws:iot:us-east-2:933819560701:topic/filter/sdk/test/java

[Builder](#) [JSON](#)

```
Received message from topic 'sdk/test/python': b'Hello World! [300]''  
Publishing message to topic 'sdk/test/python': Hello World! [301]  
Received message from topic 'sdk/test/python': b'Hello World! [301]''  
Publishing message to topic 'sdk/test/python': Hello World! [302]  
Received message from topic 'sdk/test/python': b'Hello World! [302]''  
Publishing message to topic 'sdk/test/python': Hello World! [303]  
Received message from topic 'sdk/test/python': b'Hello World! [303]''  
Publishing message to topic 'sdk/test/python': Hello World! [304]  
Received message from topic 'sdk/test/python': b'Hello World! [304]''  
Publishing message to topic 'sdk/test/python': Hello World! [305]  
Received message from topic 'sdk/test/python': b'Hello World! [305]''  
Publishing message to topic 'sdk/test/python': Hello World! [306]  
Received message from topic 'sdk/test/python': b'Hello World! [306]''  
^CTraceback (most recent call last):  
  File "/Users/saisrivatsat/Downloads/Portfolio/Projects/AWS Projects/aws-iot-device-sdk-python-v2/samples/pubsub.py", line 142, in <module>  
    time.sleep(1)  
KeyboardInterrupt  
  
(base) saisrivatsat@Sanjus-Air AWS Projects % ls -l  
total 64  
drwxr-xr-x  2 saisrivatsat  staff   64 Sep  8 02:14 IOT  
-rw-r--r--@  1 saisrivatsat  staff  1053 Sep  8 2024 IoTProject-Policy  
-rw-r--r--@  1 saisrivatsat  staff  1220 Sep  8 2024 IoTProject.cert.pem  
-rw-r--r--@  1 saisrivatsat  staff  1675 Sep  8 2024 IoTProject.private.key  
-rw-r--r--@  1 saisrivatsat  staff   451 Sep  8 2024 IoTProject.public.key  
drwxr-xr-x  6 saisrivatsat  staff  192 Aug 23 00:47 Spotify End to End Project  
drwxr-xr-x  22 saisrivatsat  staff   704 Sep  8 02:23 aws-iot-device-sdk-python-v2  
-rw-r--r--@  1 saisrivatsat  staff  6329 Sep  8 02:20 connect_device_package.zip  
-rw-r--r--  1 saisrivatsat  staff  1188 Sep  8 02:23 root-CA.crt  
-rwxr-xr-x@ 1 saisrivatsat  staff  1354 Sep  8 2024 start.sh  
(base) saisrivatsat@Sanjus-Air AWS Projects %
```

```
Publishing message to topic 'sdk/test/python': Hello World! [302]  
Received message from topic 'sdk/test/python': b'Hello World! [302]''  
Publishing message to topic 'sdk/test/python': Hello World! [303]  
Received message from topic 'sdk/test/python': b'Hello World! [303]''  
Publishing message to topic 'sdk/test/python': Hello World! [304]  
Received message from topic 'sdk/test/python': b'Hello World! [304]''  
Publishing message to topic 'sdk/test/python': Hello World! [305]  
Received message from topic 'sdk/test/python': b'Hello World! [305]''  
Publishing message to topic 'sdk/test/python': Hello World! [306]  
Received message from topic 'sdk/test/python': b'Hello World! [306]''  
^CTraceback (most recent call last):  
  File "/Users/saisrivatsat/Downloads/Portfolio/Projects/AWS Projects/aws-iot-device-sdk-python-v2/samples/pubsub.py", line 142, in <module>  
    time.sleep(1)  
KeyboardInterrupt  
  
(base) saisrivatsat@Sanjus-Air AWS Projects % ls -l  
total 64  
drwxr-xr-x  2 saisrivatsat  staff   64 Sep  8 02:14 IOT  
-rw-r--r--@  1 saisrivatsat  staff  1053 Sep  8 2024 IoTProject-Policy  
-rw-r--r--@  1 saisrivatsat  staff  1220 Sep  8 2024 IoTProject.cert.pem  
-rw-r--r--@  1 saisrivatsat  staff  1675 Sep  8 2024 IoTProject.private.key  
-rw-r--r--@  1 saisrivatsat  staff   451 Sep  8 2024 IoTProject.public.key  
drwxr-xr-x  6 saisrivatsat  staff  192 Aug 23 00:47 Spotify End to End Project  
drwxr-xr-x  22 saisrivatsat  staff   704 Sep  8 02:23 aws-iot-device-sdk-python-v2  
-rw-r--r--@  1 saisrivatsat  staff  6329 Sep  8 02:20 connect_device_package.zip  
-rw-r--r--  1 saisrivatsat  staff  1188 Sep  8 02:23 root-CA.crt  
-rwxr-xr-x@ 1 saisrivatsat  staff  1354 Sep  8 2024 start.sh  
(base) saisrivatsat@Sanjus-Air AWS Projects % code .
```

## Modify the Sample Client

### 1. Open the Python Script:

- Open the `pubsub.py` file from the connection kit.

```

$ start.sh
1  # Check to see if Root CA file exists, download if not
2  if [ ! -f ./root-CA.crt ]; then
3      printf "\nDownloading AWS IoT Root CA certificate from AWS...\n"
4      curl https://www.amazontrust.com/repository/AmazonRootCA1.pem > root-CA.crt
5  fi
6
7  # Check to see if AWS Device SDK for Python exists, download if not
8  if [ ! -d ./aws-iot-device-sdk-python-v2 ] ; then
9      printf "\nCloning the AWS SDK...\n"
10     git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git --recursive
11  fi
12
13  # Check to see if AWS Device SDK for Python is already installed, install if not
14  if ! python3 -c "import awsiot" &> /dev/null; then
15      printf "\nInstalling AWS SDK...\n"
16      python3 -m pip install ./aws-iot-device-sdk-python-v2
17      result=$?
18      if [ $result -ne 0 ]; then
19          printf "\nERROR: Failed to install SDK.\n"
20          exit $result
21      fi
22
23  # run pub/sub sample app using certificates downloaded in package
24  print "Running pub/sub sample application...\n"
25  python3 aws-iot-device-sdk-python-v2/samples/pubsub.py --endpoint axt9zjbrwtpo-ats.iot.us-east-2...

```

## 2. Add JSON-formatted Payload:

Modify the script to send CPU utilization data in a JSON format:

```

aws-iot-device-sdk-python-v2 > samples > pubsub.py > ...
125     # This step is skipped if message is blank.
126     # This step loops forever if count was set to 0.
127     if message_string:
128         if message_count == 0:
129             print("Sending messages until program killed")
130         else:
131             print("Sending {} message(s)".format(message_count))
132
133         publish_count = 1
134         while (publish_count <= message_count) or (message_count == 0):
135             message = "{} {}".format(message_string, publish_count)
136             print("Publishing message to topic '{}' : {}".format(message_topic, message))
137             message_json = json.dumps(message)
138             mqtt_connection.publish(
139                 topic=message_topic,
140                 payload=message_json,
141                 qos=mqtt.QoS.AT_LEAST_ONCE)
142             time.sleep(1)
143             publish_count += 1
144
145         # Wait for all messages to be received.
146         # This waits forever if count was set to 0.
147         if message_count != 0 and not received_all_event.is_set():
148             print("Waiting for all messages to be received...")
149
150         received_all_event.wait()
151         print("{} message(s) received.".format(received_count))
152
153     # Disconnect
154     print("Disconnecting...")
155     disconnect_future = mqtt_connection.disconnect()
156     disconnect_future.result()
157     print("Disconnected!")

```

The screenshot shows the Robocoder IDE interface. The left sidebar displays the project structure under 'AWS PROJECTS' for 'aws-iot-device-sdk-python-v2'. The 'samples' folder contains several files: pkcs11\_connect.py, pkcs12\_connect.md, pkcs12\_connect.py, pubsub.md, and pubsub.py. The 'pubsub.py' file is currently selected and open in the main editor area. The code in 'pubsub.py' is a script for publishing messages to an MQTT topic. A code completion dropdown is open over the line of code where 'psutil' is being imported. The dropdown lists several options, with 'psutil' being the top suggestion.

```
125 # This step is skipped if message is blank.
126 # This step loops forever if count was set to 0.
127 if message_string:
128     if message_count == 0:
129         print("Sending messages until program killed")
130     else:
131         print("Sending {} message(s)".format(message_count))
132
133 publish_count = 1
134 while (publish_count <= message_count) or (message_count == 0):
135     # message = "{} [{}], format(message_string, publish_count)
136     # print("Publishing message to topic '{}': {}".format(message_topic, message))
137     # message_json = json.dumps(message)
138     mqtt_connection.publish(
139         topic=message_topic,
140         payload=message_json,
141         qos=mqtt.QoS.AT_LEAST_ONCE)
142     time.sleep(1)
143     publish_count += 1
144
145 # Wait for all messages to be received.
146 # This waits forever if count was set to 0.
147 if message_count != 0 and not received_all_event.is_set():
148     print("Waiting for all messages to be received...")
149
150 received_all_event.wait()
151 print("{} message(s) received.".format(received_count))
152
153 # Disconnect
154 print("Disconnecting...")
155 disconnect_future = mqtt_connection.disconnect()
156 disconnect_future.result()
157 print("Disconnected!")
```

This screenshot shows the same Robocoder IDE environment as the previous one, but with a different code completion dropdown. The 'pubsub.py' file is still open, and the completion dropdown is now focused on the 'message\_json' variable, which is being used to construct a JSON payload. The suggestions listed include 'time', 'quality', 'hostname', and 'value', each followed by a description and a preview of the JSON structure it would create.

```
125 # This step is skipped if message is blank.
126 # This step loops forever if count was set to 0.
127 if message_string:
128     if message_count == 0:
129         print("Sending messages until program killed")
130     else:
131         print("Sending {} message(s)".format(message_count))
132
133 publish_count = 1
134 while (publish_count <= message_count) or (message_count == 0):
135     # message = "{} [{}], format(message_string, publish_count)
136     # print("Publishing message to topic '{}': {}".format(message_topic, message))
137     # message_json = json.dumps(message)
138     import psutil
139     message_json = json.dumps([
140         {
141             "time": int(time.time()),
142             "quality": "GOOD",
143             "hostname": "Laptop",
144             "value": psutil.cpu_percent(),
145         }, indent=2]
146     )
147     mqtt_connection.publish(
148         topic=message_topic,
149         payload=message_json,
150         qos=mqtt.QoS.AT_LEAST_ONCE)
151     time.sleep(1)
152     publish_count += 1
153
154 # Wait for all messages to be received.
155 # This waits forever if count was set to 0.
156 if message_count != 0 and not received_all_event.is_set():
157     print("Waiting for all messages to be received...")
158
159 received_all_event.wait()
160 print("{} message(s) received.".format(received_count))
```

```
aws-iot-device-sdk-python-v2 > samples > pubsub.py > ...  
125     # This step is skipped if message is blank.  
126     # This step loops forever if count was set to 0.  
127  
128     if message_string:  
129         if message_count == 0:  
130             print("Sending messages until program killed")  
131         else:  
132             print("Sending {} message(s).format(message_count))  
133  
134         publish_count = 1  
135         while (publish_count <= message_count) or (message_count == 0):  
136             # message = "{} [{}])".format(message_string, publish_count)  
137             # print("Publishing message to topic '{}': {}".format(message_topic, message))  
138             # message_json = json.dumps(message)  
139             import psutil  
140             message_json = json.dumps(  
141                 {  
142                     "time": int(time.time()),  
143                     "quality": "GOOD",  
144                     "hostname": "Laptop",  
145                     "value": psutil.cpu_percent(),  
146                 }, indent=2)  
147             mqtt_connection.publish(  
148                 topic=message_topic,  
149                 payload=message_json,  
150                 qos=mqtt.QoS.AT_LEAST_ONCE)  
151             print(message_json)  
152             time.sleep(1)  
153             publish_count += 1  
154  
155             # Wait for all messages to be received.  
156             # This waits forever if count was set to 0.  
157             if message_count != 0 and not received_all_event.is_set():  
158                 print("Waiting for all messages to be received...")  
159             received_all_event.wait()  
160             print("{} message(s) received.".format(received_count))
```

- Run ./start.sh on your terminal this will start sending data

```
[(base) saisrivatsat@Sanjus-Air AWS Projects % code .
zsh: command not found: code
[(base) saisrivatsat@Sanjus-Air AWS Projects % code .
[(base) saisrivatsat@Sanjus-Air AWS Projects % ./start.sh

Running pub/sub sample application...
Connecting to axt9zjbrwtipo-ats.iot.us-east-2.amazonaws.com with client ID 'basicPubSub'...
Connection Successful with return code: 0 session present: True
Connected!
Subscribing to topic 'sdk/test/python'...
Subscribed with 1
Sending messages until program killed
{
    "time": 1725781629,
    "quality": "GOOD",
    "hostname": "Laptop",
    "value": 0.0
}
Received message from topic 'sdk/test/python': b'{\n    "time": 1725781629,\n    "quality": "GOOD",\n    "hostname": "Laptop",\n    "value": 0.0\n}'
{
    "time": 1725781630,
    "quality": "GOOD",
    "hostname": "Laptop",
    "value": 29.8
}
Received message from topic 'sdk/test/python': b'{\n    "time": 1725781630,\n    "quality": "GOOD",\n    "hostname": "Laptop",\n    "value": 29.8\n}'
{
    "time": 1725781631,
    "quality": "GOOD",
    "hostname": "Laptop",
    "value": 20.8
}
```

```
"time": 1725781629,
"quality": "GOOD",
"hostname": "Laptop",
"value": 0.0
}
Received message from topic 'sdk/test/python': b'{\n    "time": 1725781629,\n    "quality": "GOOD",\n    "hostname": "Laptop",\n    "value": 0.0\n}'
{
    "time": 1725781630,
    "quality": "GOOD",
    "hostname": "Laptop",
    "value": 29.8
}
Received message from topic 'sdk/test/python': b'{\n    "time": 1725781630,\n    "quality": "GOOD",\n    "hostname": "Laptop",\n    "value": 29.8\n}'
{
    "time": 1725781631,
    "quality": "GOOD",
    "hostname": "Laptop",
    "value": 20.8
}
Received message from topic 'sdk/test/python': b'{\n    "time": 1725781631,\n    "quality": "GOOD",\n    "hostname": "Laptop",\n    "value": 20.8\n}'
{
    "time": 1725781632,
    "quality": "GOOD",
    "hostname": "Laptop",
    "value": 25.4
}
Received message from topic 'sdk/test/python': b'{\n    "time": 1725781632,\n    "quality": "GOOD",\n    "hostname": "Laptop",\n    "value": 25.4\n}'
```

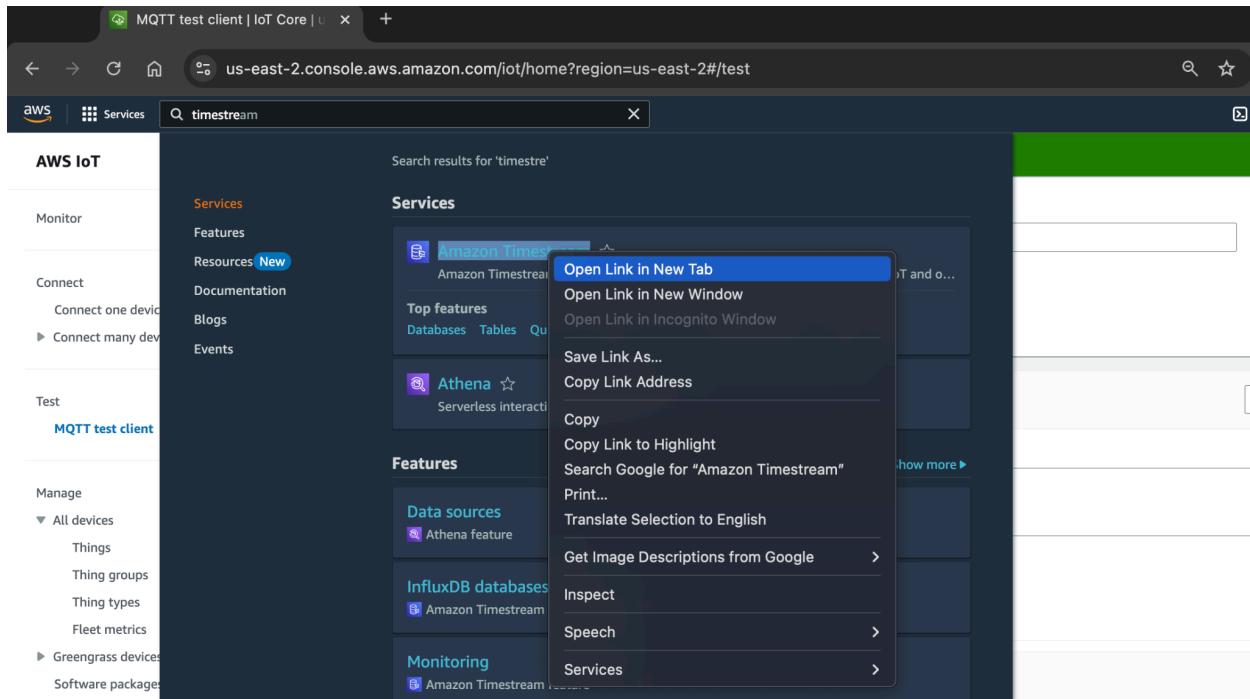
The screenshot shows the AWS IoT MQTT test client interface. At the top, a green banner indicates "AWS IoT successfully created thing resource IoTProject and generated your connection kit." Below this, the left sidebar includes sections for Monitor, Connect (with options for one device or many), and Test (with an MQTT test client option). The main content area shows a "Subscriptions" section for the topic "sdk/test/python". It displays a "Message payload" with the JSON content: { "message": "Hello from AWS IoT console" }. There are "Additional configuration" and "Publish" buttons. Below this, two message entries are listed under "sdk/test/python": one from "September 08, 2024, 02:48:56 (UTC-0500)" and another from "September 08, 2024, 02:48:55 (UTC-0500)". Each message entry has a "Properties" button.

## Storing Data in Amazon Timestream

### Create a Timestream Database

#### 1. Open Amazon Timestream:

- In the AWS Console, search for **Amazon Timestream** and open it.



## 2. Create a New Database:

- Click **Create Database**, provide a name (e.g., **IoTDataDB**), and accept the default encryption settings.

**Amazon Timestream**  
Easy to manage time-series databases optimized for security, performance, availability, and scalability

**Get started with Timestream**

**Timestream for LiveAnalytics**  
With Timestream for LiveAnalytics, you can ingest more than tens of gigabytes of time-series data per minute and run SQL queries on terabytes of time-series data in seconds with up to 99.99% availability. It has built-in time-series analytics functions, helping you identify trends and patterns in near real time.

**Timestream for InfluxDB**  
With Amazon Timestream for InfluxDB, you can easily run open-source InfluxDB databases on AWS for time-series applications with millisecond response times. Timestream for InfluxDB provides up to 99.9% availability. Ideal use cases for Timestream for InfluxDB include real-time alerting and monitoring infrastructure reliability.

**AWS Free Tier for Timestream for LiveAnalytics**  
For new accounts, explore the power of Amazon Timestream for LiveAnalytics free of charge, for 1 month. [Learn more](#)

**Pricing**  
[Pricing guide](#)

**Create database [Info](#)**

**Database configuration**  
Create and configure a database or create a database with sample data to explore Timestream right away.

**Choose a configuration**

- Standard database  
Create a new database with custom configuration.
- Sample database  
Create a database and populate it with sample data to get started in a single click.

**Name**  
Specify a name that is unique for all Timestream databases in your AWS account in the current Region. You can not change this name once you create it.  
**IoTProjectDB**  
Must be between 3 and 256 characters long. Must contain letters, digits, dashes, periods or underscores.

**Encryption**  
All Amazon Timestream data is encrypted by default.

**KMS key**  
KMS key IDs and aliases appear in the list after they have been created using the Key Management Service (KMS) console.  
**aws/timestream** [Manage keys](#)

**Description**  
Default KMS key that protects Timestream data when no other key is defined. The key will be created by Timestream on your behalf.

**Key ARN**  
[-](#)

**Databases (1) [Info](#)**

Name	Creation time (UTC)
<a href="#">IoTProjectDB</a>	9/8/2024, 7:52:24 AM

[Create table](#)

### 3. Create a Table:

- After the database is created, click **Create Table**.
- Name the table (e.g., **IoTProjectTable**), and configure the memory/magnetic storage options for storing time-series data.

AWS Services Search [Option+S] Ohio Sanju @ 9338-1956-0701

### Amazon Timestream

- Service dashboard
- LiveAnalytics**
  - Admin dashboard
- Resources**
  - Databases
    - Tables
    - Backups
- Management Tools**
  - Batch load tasks
  - EventBridge Pipes  New
  - Query editor
  - Scheduled queries
  - Monitoring
- InfluxDB**
  - InfluxDB databases  New
  - Parameter groups
- Getting started
- Tutorials

**IoTProjectDB**

**Summary**

Name IoTProjectDB	Modified time (UTC) 9/8/2024, 7:52:24 AM	Creation time (UTC) 9/8/2024, 7:52:24 AM
Database ARN <code>arn:aws:timestream:us-east-2:933819560701:database/IoTProjectDB</code>	KMS key ARN <code>arn:aws:kms:us-east-2:933819560701:key/ecedc4e4-fcb4-4cd9-bcb5-33aa186e71b6</code>	

Monitoring Tables Tags

**Tables (0) Info**

Create backup Create scheduled query Edit Delete **Create table**

No tables

Tables you create will appear here. A database is required to create a table.

**Create table**

☰ Timestream > Databases > IoTProjectDB > Tables > Create table

### Create table Info

**Table details**

**Database name**  
Choose the database where this table will be created.  
IoTProjectDB  C

**Table name**  
Specify a table name that is unique within this database. You can not change this name once you create it.  
IoTProjectTable   
Must be between 3 and 256 characters long. Must contain letters, digits, dashes, periods or underscores.

**Schema settings -new Info**  
A Schema specifies the expected data model of the table. Specify how your data will be stored and distributed using partition keys. With custom partition keys, you can run queries that are faster and more efficient.

**Partition key configuration**  
Choose to create a custom partition key or use Timestream's default partition key.

**Custom partitioning**  
This is useful when the primary access pattern of the queries is filtering over a high cardinality dimension or measure.

**Default partitioning**  
This is useful when the primary access pattern of queries is measure based filtering. The values of 'measure\_name' attribute is used as the partition key to partition the data.

**Composite partition key Info**  
A list of partition keys defining the attributes used to partition the table data. The order of the list determines the partition hierarchy.

**1 - Partition key type**  
Choose an attribute type  Partition key name example-key

**Enforce partition key in record**

AWS Services Search [Option+S] X | A

**Memory store retention**  
Specify how long data will be stored in the memory store before it is moved to magnetic store.

1	Hour(s)
---	---------

The value must be a number. Minimum 1 hour, maximum 12 months.

**Magnetic store retention**  
Specify how long data will be stored in magnetic store before it is deleted. If this value is less than the original magnetic restore retention period, data will be lost.

1	Year(s)
---	---------

The value must be a number. Minimum 1 day, maximum 200 years.

**Magnetic Storage Writes**

Settings  Enable magnetic storage writes

**Backup settings - new [info](#)**  
Timestream integrates with AWS Backup to protect your data. Backups are incremental after the first backup is created.

Turn on backups for this table  
The table will be protected by AWS Backups until this feature is turned off.

**Tags - optional**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags are associated with this table.

[Add new tag](#)  
You can add 50 more tag(s).

[Cancel](#) [Create table](#)

## Create an IoT Core Rule:

- In the AWS IoT Core Console, go to Message routing > Rules > Create a Rule.

AWS Services Search [Option+S] X | A | O | Ohio | Sanju @ 9338-1956-0701

**AWS IoT** X

AWS IoT successfully created thing resource IoTProject and generated your connection kit.

[AWS IoT](#) > [Message routing](#) > Rules

**Rules (0) [info](#)**  
Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

<a href="#">Q. Find rules</a>	< 1 > <span style="float: right;">@</span>
No rules You don't have any rules in us-east-2.	<a href="#">Create rule</a>

Monitor

Connect

- Connect one device
- Connect many devices

Test

MQTT test client

Manage

- All devices
- Things
- Thing groups
- Thing types
- Fleet metrics
- Greengrass devices
- Software packages [New](#)
- Remote actions
- Message routing
- Rules**
- Destinations
- Retained messages
- Security
- Fleet Hub

- Name your rule (e.g., **IoTProjectToTimestream**).

AWS IoT successfully created thing resource `iotProject` and generated your connection kit.

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**

**Rule name**  
`IoTPorjectToTimestream`

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

**Rule description - optional**  
Enter a description to provide additional details about the rule to others.  
`A description of your new rule`

**Tags - optional**  
No tags associated with the resource.

**Add new tag**  
You can add up to 50 tags.

Cancel **Next**

## Define SQL Statement:

- In the SQL editor, select the incoming MQTT messages from your topic (e.g., **SDK/test/python**):

AWS IoT successfully created thing resource `iotProject` and generated your connection kit.

Step 1  
[Specify rule properties](#)

Step 2  
**Configure SQL statement**

Step 3  
Attach rule actions

Step 4  
Review and create

**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.  
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](#).

```
1 SELECT * FROM 'sdk/test/python'
```

SQL Ln 1, Col 32 Errors: 0 Warnings: 0 ⚙

Cancel Previous Next

## Configure Rule Action:

- Set the rule action to route the data to **Amazon Timestream**.

## Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

### Action 1

▼ **Timestream table**  
Write a message into a Timestream table

**Remove**

Database name [Info](#)

IoTProjectDB



[View](#)

[Create Timestream database](#)

Table name

IoTProjectTable



[View](#)

[Create Timestream table](#)

Dimensions

Each record contains an array of dimensions (minimum 1). Dimensions represent the metadata attributes of a time series data point.

Dimensions name

Dimension value

host

`#{hostname}`

**Remove**

[Add new dimension](#)

Timestamp value - *optional*

Timestamp unit

`#{time}`

SECONDS



IAM role

Choose a role to grant AWS IoT access to your endpoint.

IoTProjectToTimestreamRole



[View](#)

[Create new role](#)

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

[Add rule action](#)

- Assign an **IAM role** that allows the rule to write data to the Timestream database.

Dimensions

Each record contains an array of dimensions (minimum 1). Dimensions represent the metadata attributes of a time series data point.

Dimensions name Dimension value

host	
<a href="#">Add new dimension</a>	

Timestamp value - *optional*

#{time}
---------

IAM role

Choose a role to grant AWS IoT access

<a href="#">Choose an IAM role</a>
------------------------------------

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

Add rule action

### Create role

Role name

Enter a unique role name that contains alphanumeric characters, hyphens, and underscores. A role name can't contain any spaces.

[Cancel](#) [Create](#)

Error action - *optional*

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.

▼ Republish to AWS IoT topic Republish a message to an AWS IoT topic	<a href="#">Remove</a>
---	------------------------

Topic [Info](#)

Quality of service

When subscribing to a topic, quality of service 0 is chosen by default.

0 - The message is delivered at most once

1 - The message is delivered at least once

MQTT 5 properties [Info](#)  
Add MQTT 5 properties by choosing a property type and adding a value.

[Add property](#)  
You can add 5 more properties.

User properties [Info](#)  
Add user properties in the MQTT header.

[Add user property](#)

IAM role

Choose a role to grant AWS IoT access to your endpoint.

PublishErrors	▼	<a href="#">C</a>	<a href="#">View</a>	<a href="#">Create new role</a>
---------------	---	-------------------	----------------------	---------------------------------

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

Add error action

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

AWS Services Search [Option+S]

SELECT \* FROM 'sdk/test/python'

### Step 3: Rule actions

**Actions**

**Timestream table**  
Write a message into a Timestream table

Database name IoTProjectDB	Table name IoTProjectTable	Dimensions • 1 ◦ name: host ◦ value: \${hostname}
Timestamp value - optional • value: \${time} • unit: MILLISECONDS		IAM role <a href="#">arn:aws:iam::933819560701:role/service-role/IoTProjectToTimestreamRole</a>

**Error action**

**Republish to AWS IoT topic**  
Republish a message to an AWS IoT topic

Topic errors	Quality of service 0	IAM role <a href="#">arn:aws:iam::933819560701:role/service-role/PublishErrors</a>
MQTT 5 properties	User properties	-

Cancel Previous Create

AWS Services Search [Option+S]

Successfully created rule IoTProjectToTimestream.

AWS IoT > Message routing > Rules

**Rules (1) Info**

Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

Name	Status	Rule topic	Created date
IoTProjectToTimestream	Active	sdk/test/python	September 08, 2024, 03:08:32 (UTC-05:00)

View rule

The screenshot shows the AWS Amazon Timestream Query Editor interface. On the left, there's a navigation sidebar with sections for Service dashboard, LiveAnalytics (Admin dashboard, Resources, Management Tools), and InfluxDB (InfluxDB databases, Parameter groups). The main area has tabs for Editor, Recent, Saved queries, and Sample queries. Under Editor, a Database dropdown is set to 'IoTProjectDB'. A 'Tables (1)' section shows 'IoTPProjectTable'. The query editor contains the following SQL:

```

1 select *
2 from IoTProjectDB."IoTPProjectTable"
3 where measure_name = 'value'

```

The 'Output' tab shows the results of the query:

Start	Status	Response	Statement
3:10:59 AM	Success	0 rows affected.	select * from IoTProjectDB."IoTPProjectTable" where measure_name = 'value'

A message at the top right says 'Try scheduling your queries for improved real time analytics with cost savings.' with a 'Create scheduled query' button.

This screenshot is similar to the one above, but the 'Output' tab is replaced by the 'Query results' tab. The results show 7 rows returned from the 'IoTPProjectTable' in the 'IoTProjectDB' database. The table structure is:

host	measure_name	time	measure_value::varchar	measure_value::double
Laptop	value	2024-09-08 08:16:31.000000000	-	27.2
Laptop	value	2024-09-08 08:16:32.000000000	-	20.1
Laptop	value	2024-09-08 08:16:33.000000000	-	18.0
Laptop	value	2024-09-08 08:16:34.000000000	-	21.9

### 3.4 Visualizing Data with Amazon Managed Grafana

1. Set up a new Amazon Managed Grafana workspace.

AWS Services Search [Option+Shift] Management & Governance

# Amazon Managed Grafana

## Fully managed Grafana service with powerful, interactive data visualizations

Amazon Managed Grafana is a fully managed Grafana service that is scalable, secure, and highly available. Using Amazon Managed Grafana, you can analyze, monitor, and alarm on metrics, logs, and traces across multiple data sources.

### How it works

The diagram illustrates the workflow:

- Create workspace
- Manage user access
- Contact to analyze
- Set up dashboards
- Customize 3rd party

### Benefits and features

Enjoy the power of Grafana at scale	Visualize, analyze, and correlate securely across multiple data sources
Use the powerful data visualizations you love from Grafana, without having to worry about maintaining software or infrastructure.	Amazon Managed Grafana securely and natively integrates with AWS services, without traversing the public internet, across multiple accounts and multiple regions.

Manage access to data and [Upgrade to Amazon Managed Grafana](#)

**Create workspace**  
A workspace is a logically isolated Grafana server where you can create Grafana dashboards and visualization to analyze your metrics, logs, and traces.  
[Create workspace](#)

**Getting started**  
[What Is Amazon Managed Grafana?](#)  
[Getting started with Amazon Managed Grafana](#)  
[Create a workspace](#)

**More resources**  
[Documentation](#)  
[FAQ](#)  
[Forum](#)  
[Contact us](#)

Amazon Grafana > Workspaces > Create new workspace

Step 1 Specify workspace details

Step 2 Configure settings

Step 3 Service managed permission settings

Step 4 Review and create

### Specify workspace details

A workspace is a logically isolated Grafana server. Once you have created a workspace, you can integrate it with data sources, then query and visualize metrics from those data sources. As part of creating a workspace, you will enable AWS IAM Identity Center (successor to AWS SSO) if you haven't done so already.

#### Workspace details

**Workspace name**  
Give a unique name to your workspace.  
  
Valid special characters include "., \_ , - , ~". Cannot contain non-ASCII characters or spaces. Max length of 255 characters.

**Workspace description - optional**

**Grafana version**  
Select the Grafana version you want to use for this workspace.

**Tags - optional**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.  
Currently not editing fields.

[Add new tag](#)  
You can add up to 50 more tags.

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

Amazon Grafana > Workspaces > Create new workspace

Step 1 [Specify workspace details](#)

Step 2 [Configure settings](#)

Step 3 Service managed permission settings

Step 4 Review and create

## Configure settings [Info](#)

### Authentication access [Info](#)

**Choose at least one authentication method.**

AWS IAM Identity Center (successor to AWS SSO) ⚠ Not enabled

You can enable IAM Identity Center by creating a user. This new user does not automatically have access to the Grafana console. You will still need to assign this user later, once this workspace is created.

Security Assertion Markup Language (SAML)

You will need to complete additional steps to finish SAML configuration once this workspace is created.

ⓘ For IAM Identity Center to be usable you need to add a user to the service. Note that this new user does not automatically have access to the Grafana console. You will still need to assign this user later, once this workspace is created. [Create user](#)

ⓘ The workspace needs at least one authentication method. IAM Identity Center has to be enabled before continuing. SAML can be set up after the workspace have been created.

### Permission type [Info](#)

Step 1 [Specify workspace details](#)

Step 2 [Configure settings](#)

Step 3 [Service managed permission settings](#)

Step 4 Review and create

## Service managed permission settings [Info](#)

### IAM permission access settings

Select how you would like to specify account access.

Current account  
Use Grafana to monitor resources in your current account.

Organization  
Use Grafana to monitor resources in your Organizational Units (OUs).

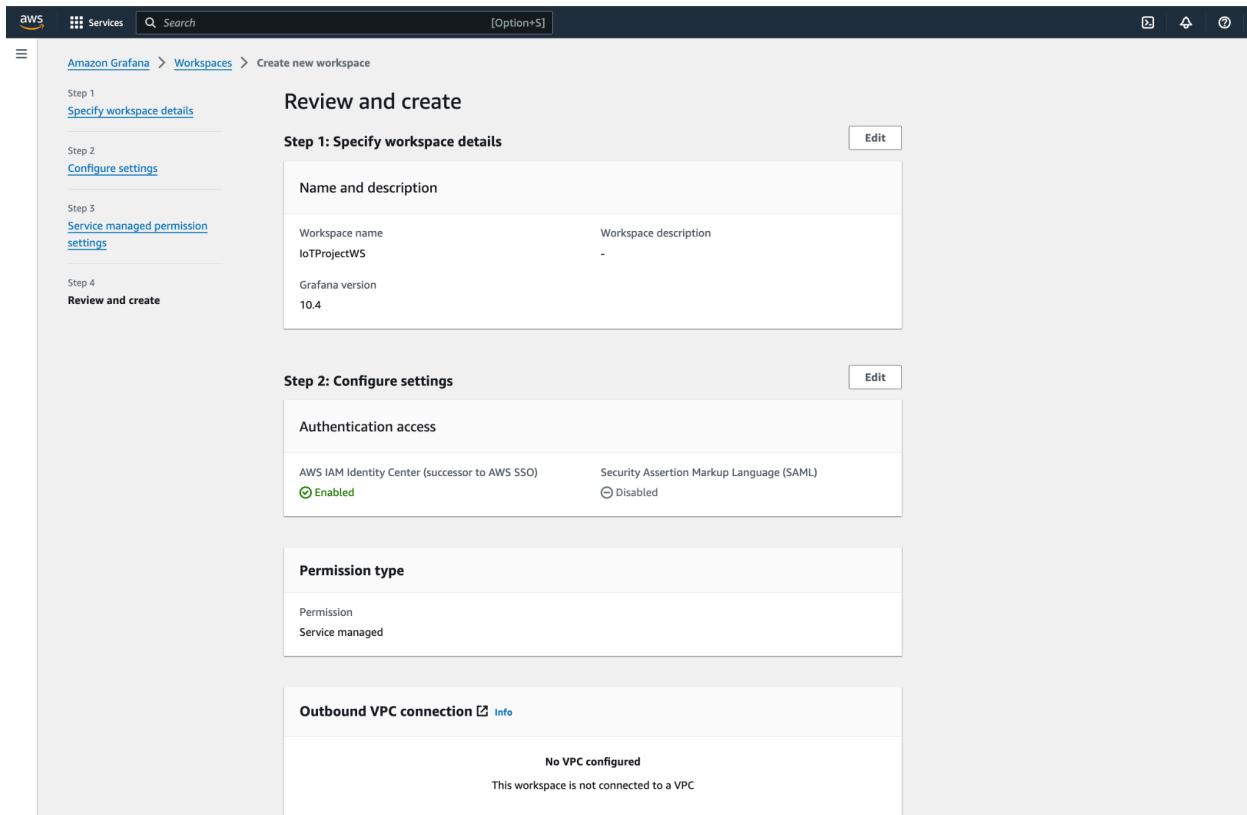
### ▼ Data sources and notification channels - optional

#### Data sources

Selecting an AWS data source below creates an IAM role that enables Amazon Grafana access to those resources in your current account. It does not set up the selected service as a data source. Note that some resources must be tagged GrafanaDataSource to be accessible.

<input type="checkbox"/> Data source name
<input type="checkbox"/> AWS IoT SiteWise
<input type="checkbox"/> AWS X-Ray
<input type="checkbox"/> Amazon CloudWatch
<input type="checkbox"/> Amazon OpenSearch Service
<input type="checkbox"/> Amazon Managed Service for Prometheus
<input checked="" type="checkbox"/> Amazon TimeStream
<input type="checkbox"/> Amazon Redshift
<input type="checkbox"/> Amazon Athena

#### Notification channels



2. Connect your Timestream database as a data source in Grafana.
  3. Create a new dashboard and panel to visualize the incoming data in real-time.
- 

## 4. Summary

In this demonstration, we created an AWS IoT Core device, routed its telemetry data to Amazon Timestream, and visualized the data using Amazon Managed Grafana. This end-to-end solution showcases how to set up a scalable, secure, and efficient IoT data pipeline using only three AWS services.

---

## 5. Additional Resources

- [AWS IoT Core Documentation](#)
- [Amazon Timestream Documentation](#)
- [Amazon Managed Grafana Documentation](#)