Cloud Backend for Remote Lora Networks

Cloud Setup Guide

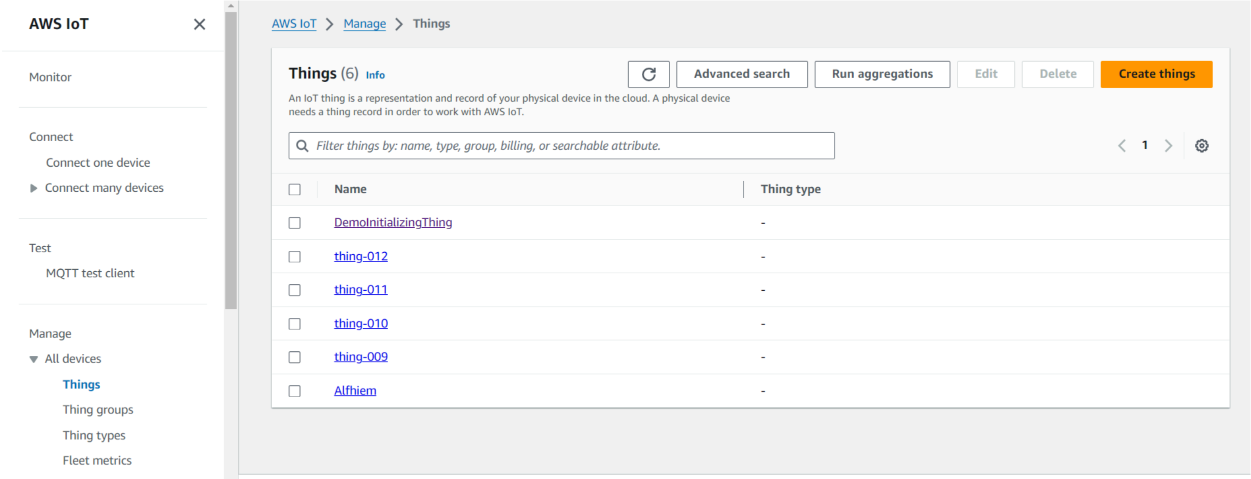
Step 1: Setting up an AWS Thing

We need to create an AWS Thing to initiate our AWS IoT integration. The name of this Thing should be recorded in our GatewayWebServerModule.cpp file before being compiled into our Gateway.

After creating the Thing, AWS generates associated certificates. These certificates serve as crucial authentication tokens for secure communication between our devices and AWS IoT services, and they should be stored in the secrets.h file provided in the GitHub repository.

Now, A Step by Step process to create an AWS IOT Thing on AWS Console:

- Launch the IoT Core service in AWS
- Click **All Devices -> Things** in the sidebar to receive a menu like seen below. And click the yellow Create Thing button.

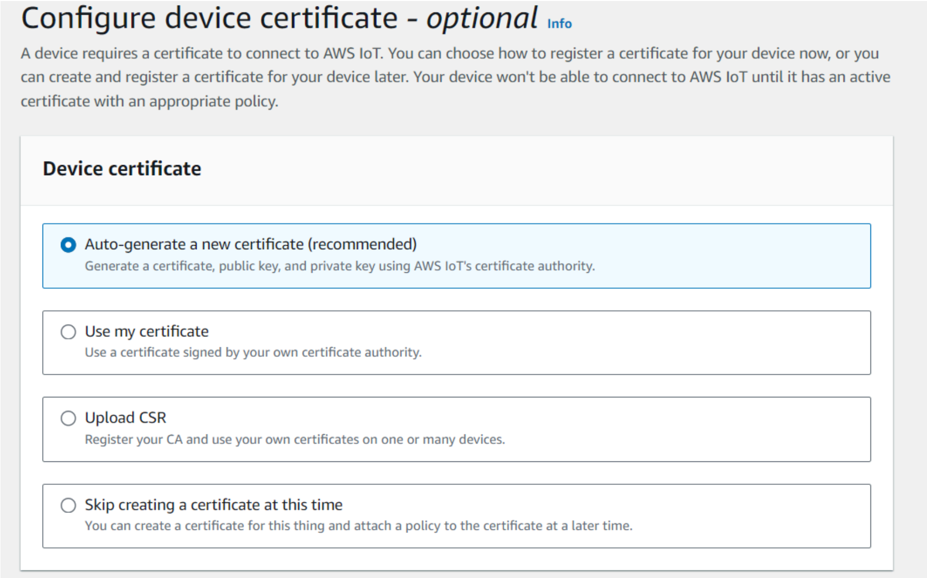


An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Name	Thing type
DemoInitializingThing	-
thing-012	-
thing-011	-
thing-010	-
thing-009	-
Alfhiem	-

- AWS will display a simple form with three small steps.
    - For Step 1: “Specify Thing Property,” set the Thing Name field to any name of choice and leave the rest of the field as its default value.

- For Step 2: "Configure Device Certificate," select the auto-generate a new certificate field, as seen in the screenshot below.



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

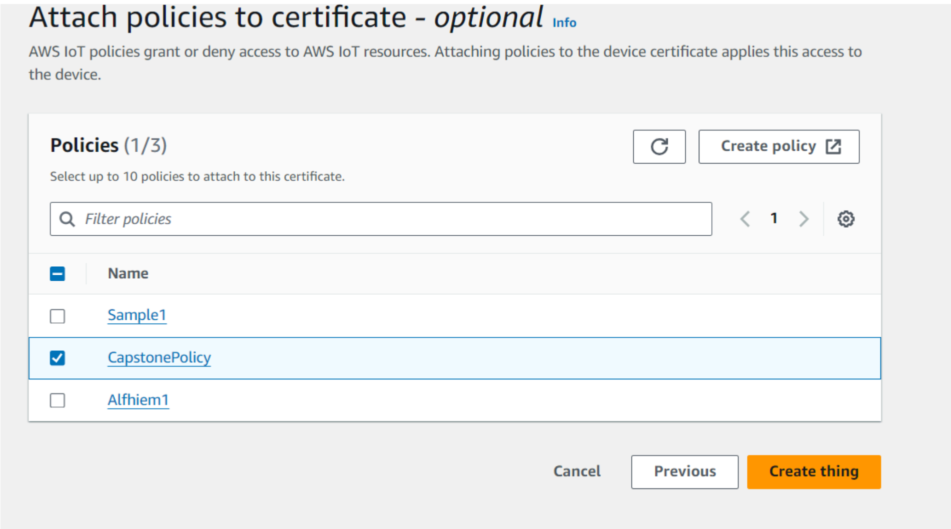
● Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

○ 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.

- For Step 3: Here, we attach the policies, which are the actions the certificate will allow the IOT device to take, like Connect, Subscribe, and Publish. In case the policy does not exist, we can create a new one by clicking on Create Policy.



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.

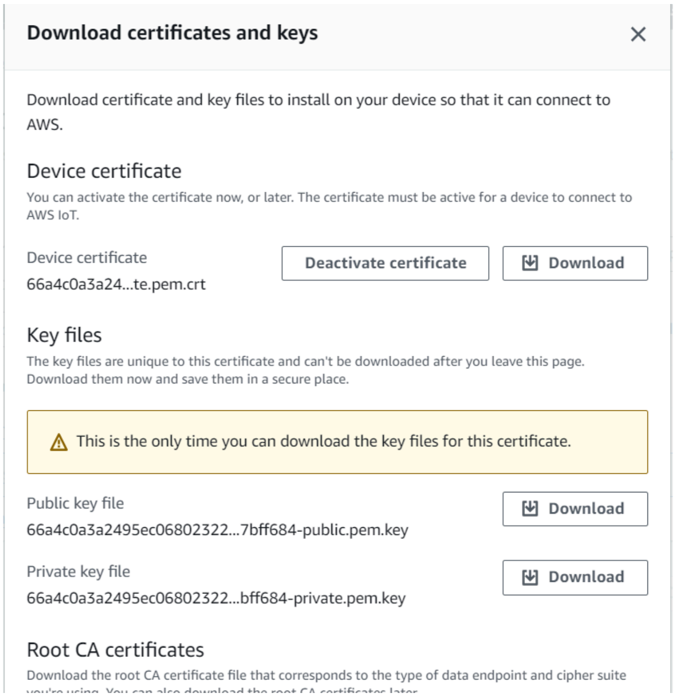
Policies (1/3)

Select up to 10 policies to attach to this certificate.

	Name
□	Sample1
☑	CapstonePolicy
□	Alfhiem1

- The configuration of the new policy should look like the following Image. Then, we can go back to the page shown in Step 3 and click on the policy created.

- Once the thing creation is successful, AWS will generate the certificates for the Thing. The next step is to download and save these certificates. The Device Certificate, The Private Key, and Root CA Certificates can be pasted into the secrets.h file so that a secure connection with appropriate credentials can be established with AWS.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
66a4c0a3a24...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

Public key file
66a4c0a3a2495ec06802322...7bff684-public.pem.key

Private key file
66a4c0a3a2495ec06802322...bff684-private.pem.key

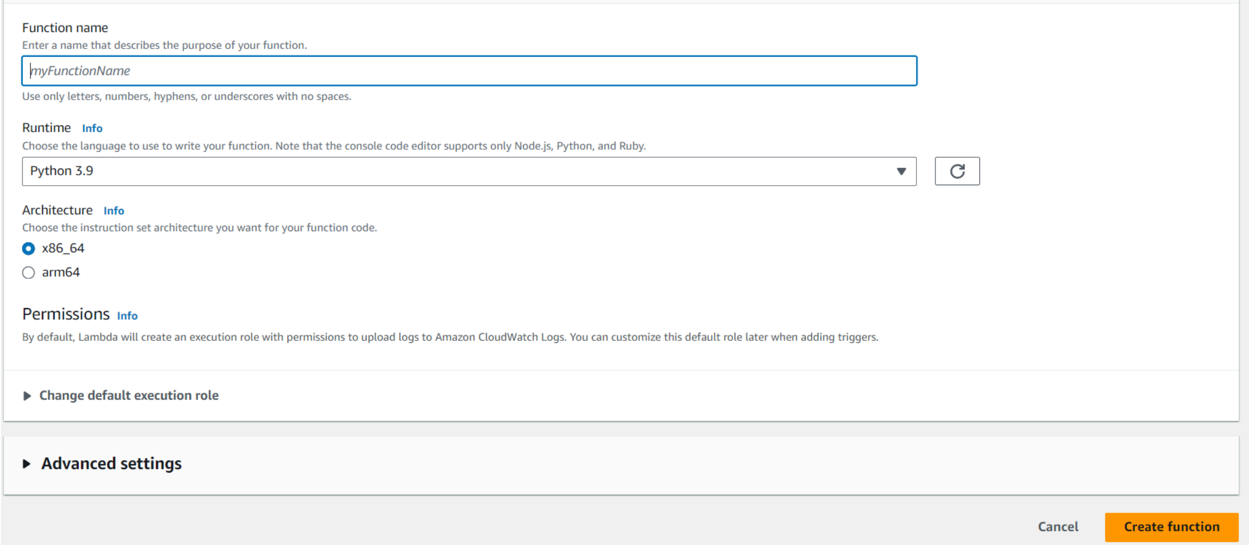
Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Step 2: Creating AWS Lambda Functions and Configuring Permissions

- To deploy our application logic to AWS Lambda, we follow these steps:
- Create Lambda Functions: We create two Lambda functions, namely GatewayInitialization and GatewayDataHandling. These functions will host the code from our GitHub repository. We can modify this code to tailor it to our specific application requirements.
- Set Permissions: Each Lambda function requires specific permissions to execute properly and interact with other AWS services. By configuring these permissions correctly, we ensure that our Lambda functions can seamlessly integrate with AWS IoT services and perform their designated tasks effectively. Additionally, this setup provides the necessary security measures to safeguard our IoT infrastructure and application logic.

- Launch the AWS Lambda service, and click on Create New Function, and follow the configurations in the following two images.



Function name

Enter a name that describes the purpose of your function.

myFunctionName

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9

Architecture Info

Choose the instruction set architecture you want for your function code.

● x86_64

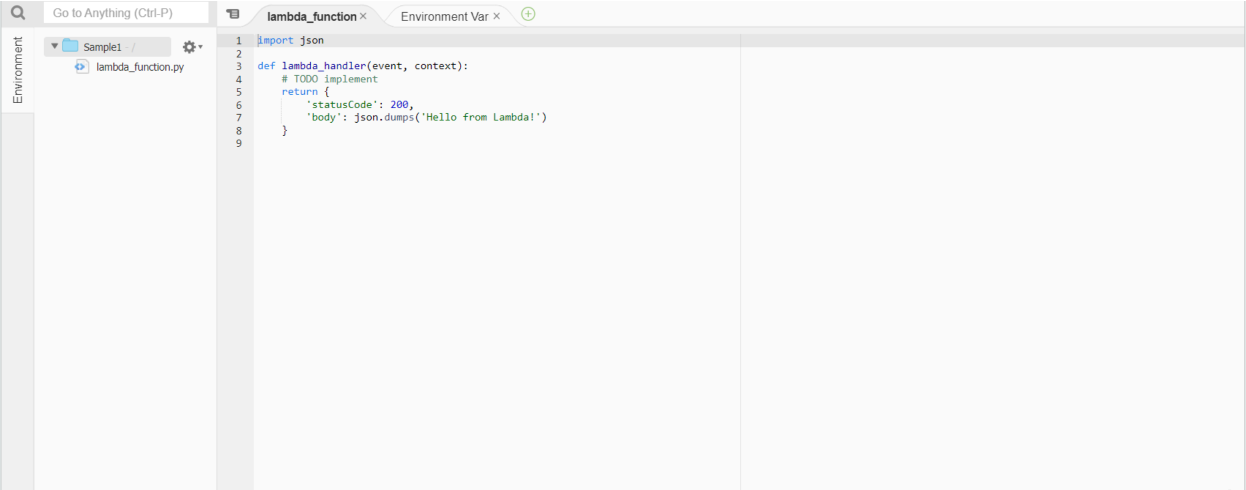
○ arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role

▶ Advanced settings

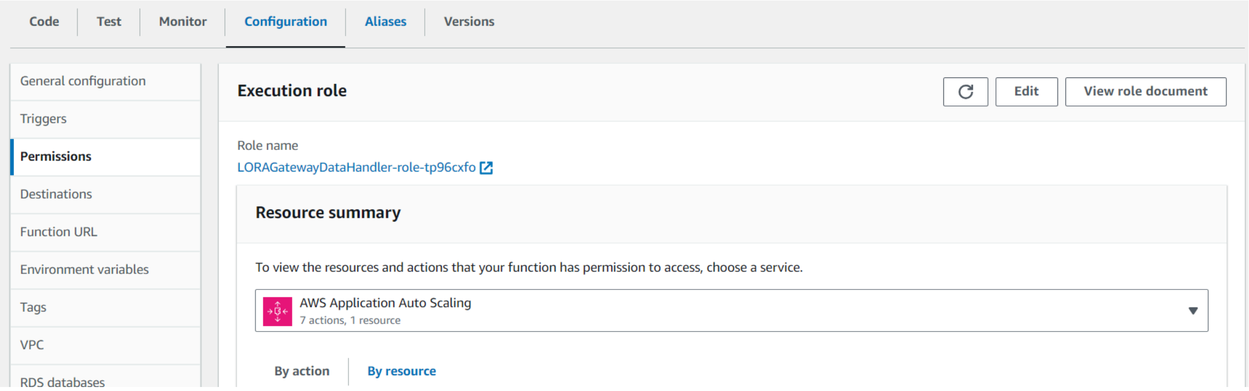


```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

- Now, transfer the code for each of the Lambda Functions.
- Now, go to Configurations, then Permissions in the Lambda Functions:



Execution role

Role name

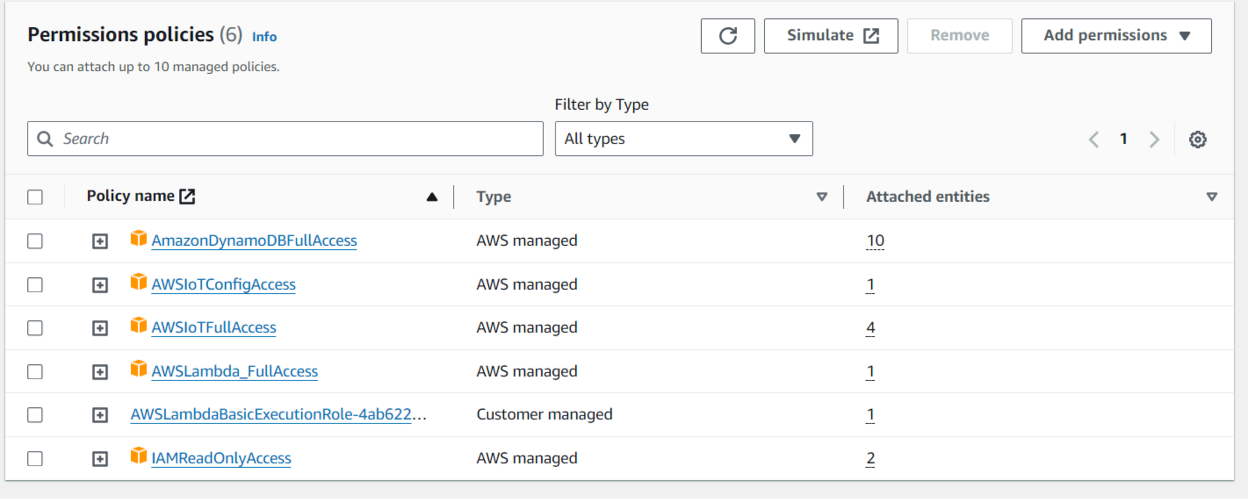
LORAGatewayDataHandler-role-tp96cxfo

Resource summary

To view the resources and actions that your function has permission to access, choose a service.

AWS Application Auto Scaling
7 actions, 1 resource

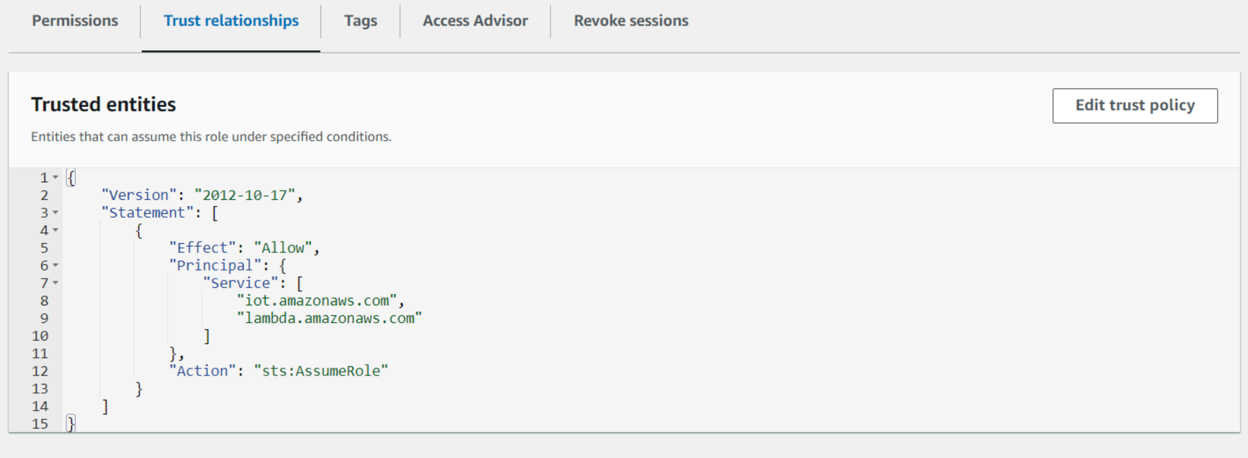
- GatewayInitialization Lambda Function Permissions and Trust Relationships



Permissions policies (6) Info

You can attach up to 10 managed policies.

□		Policy name	Type	Attached entities
□	⊞	AmazonDynamoDBFullAccess	AWS managed	10
□	⊞	AWSIoTConfigAccess	AWS managed	1
□	⊞	AWSIoTFullAccess	AWS managed	4
□	⊞	AWSLambda_FullAccess	AWS managed	1
□	⊞	AWSLambdaBasicExecutionRole-4ab622…	Customer managed	1
□	⊞	IAMReadOnlyAccess	AWS managed	2

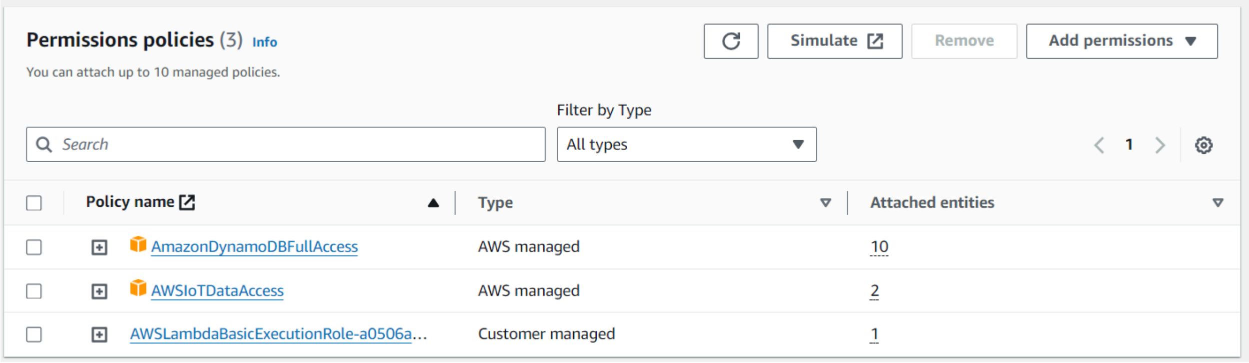


Trusted entities

Entities that can assume this role under specified conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "iot.amazonaws.com",
                    "lambda.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- GatewayDataHandling Lambda Function Permissions:



Permissions policies (3) Info

You can attach up to 10 managed policies.

□		Policy name	Type	Attached entities
□	⊞	AmazonDynamoDBFullAccess	AWS managed	10
□	⊞	AWSIoTDataAccess	AWS managed	2
□	⊞	AWSLambdaBasicExecutionRole-a0506a…	Customer managed	1

Now, we create the Rules to Link to the Gateway Initialization Lambda Function.

Step 3: Creation of an AWS IoT Rule

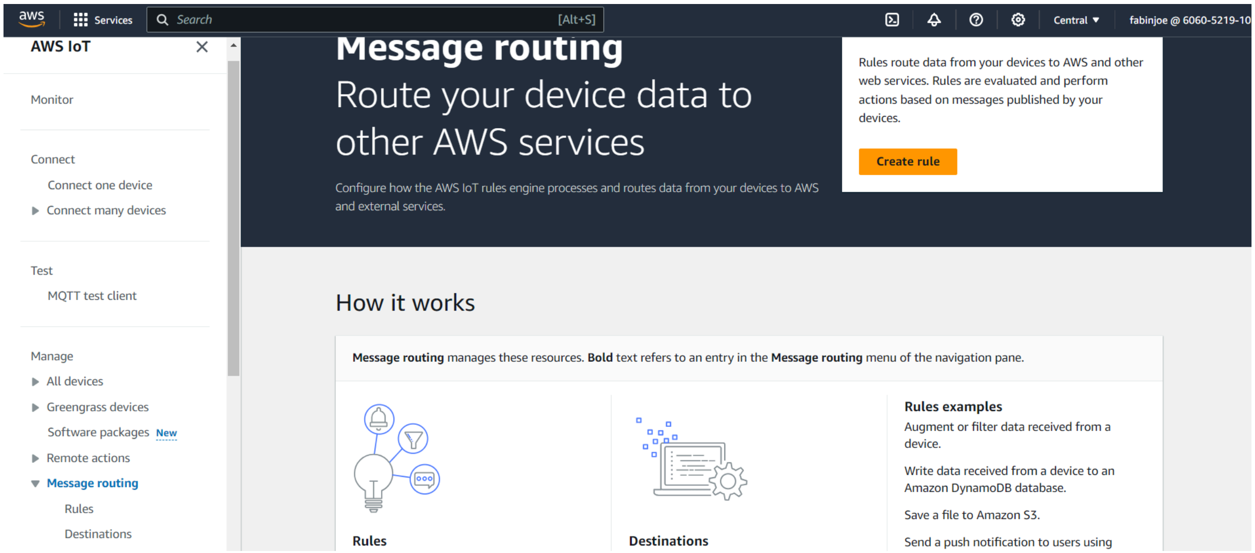
To streamline message routing within our AWS IoT ecosystem, we create an AWS IoT Rule designed to handle the initialization messages sent from all gateways. This rule ensures that these initialization messages are efficiently routed to the designated Gateway Initialization Lambda Function.

It's worth noting that the Lambda function responsible for gateway initialization will also handle the subsequent message routing for sensor data. Therefore, there's no need to create additional rules for this purpose. This setup simplifies our configuration and ensures that message routing is effectively managed within our IoT infrastructure.

Once configured, this rule only needs to be set up once, providing a seamless and automated process for handling gateway initialization messages moving forward.

The steps to do so are as follows:

- First, we go to Message Routing in AWS IOT and then create a Rule by clicking Create Rule, as shown in the image.



Message routing

Route your device data to other AWS services

Configure how the AWS IoT rules engine processes and routes data from your devices to AWS and external services.

Rules route data from your devices to AWS and other web services. Rules are evaluated and perform actions based on messages published by your devices.

How it works

Message routing manages these resources. **Bold** text refers to an entry in the **Message routing** menu of the navigation pane.

Rules examples

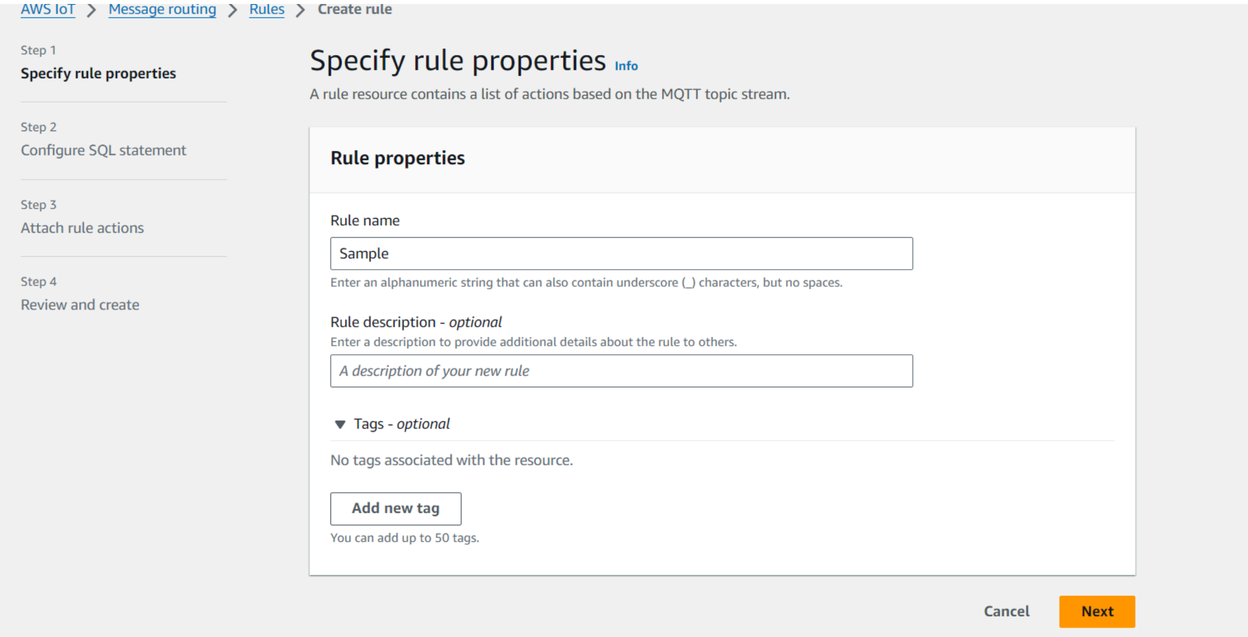
Augment or filter data received from a device.

Write data received from a device to an Amazon DynamoDB database.

Save a file to Amazon S3.

Send a push notification to users using

- We give the rule a name.



Specify rule properties

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

Rule properties

Rule name: Sample

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.

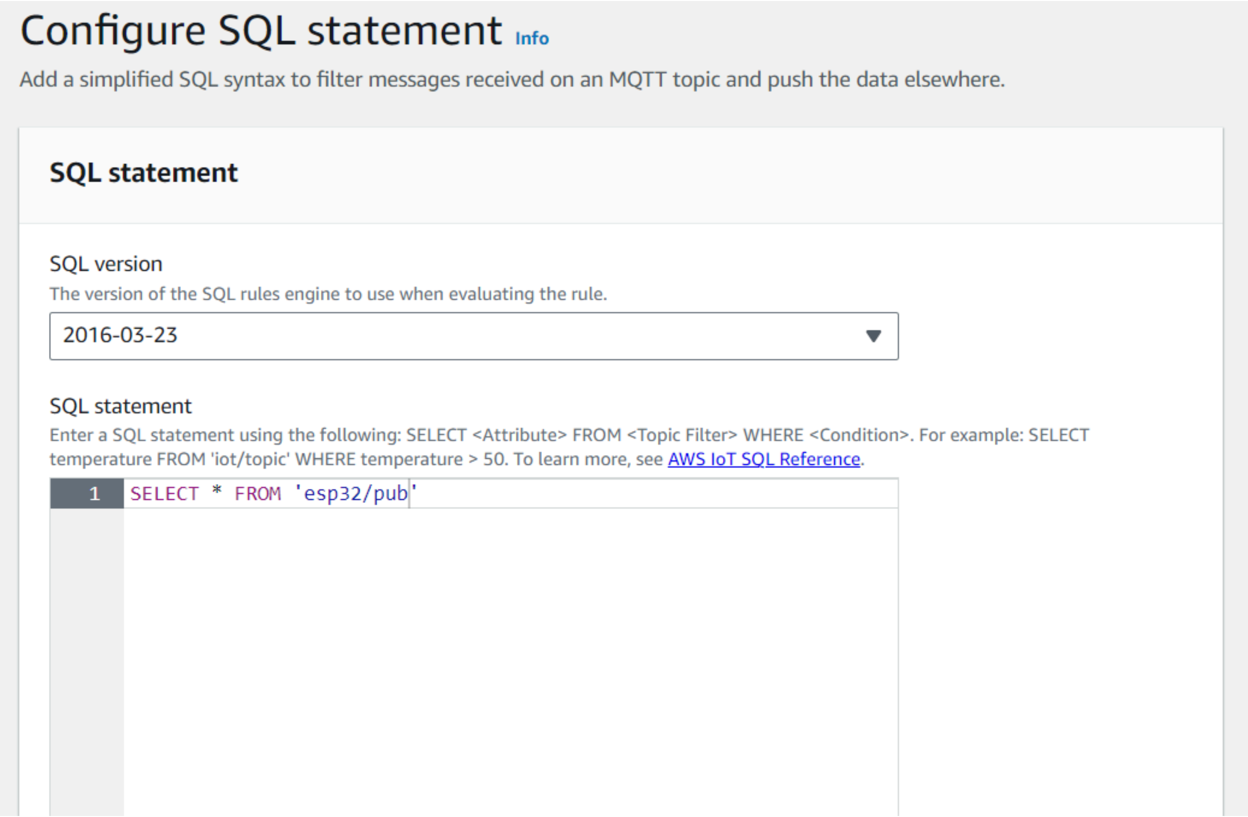
A description of your new rule

▼ Tags - *optional*

No tags associated with the resource.

You can add up to 50 tags.

- Next, we configure the topic to which all gateways will send initialization messages, as shown in the image below.



Configure SQL statement

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.

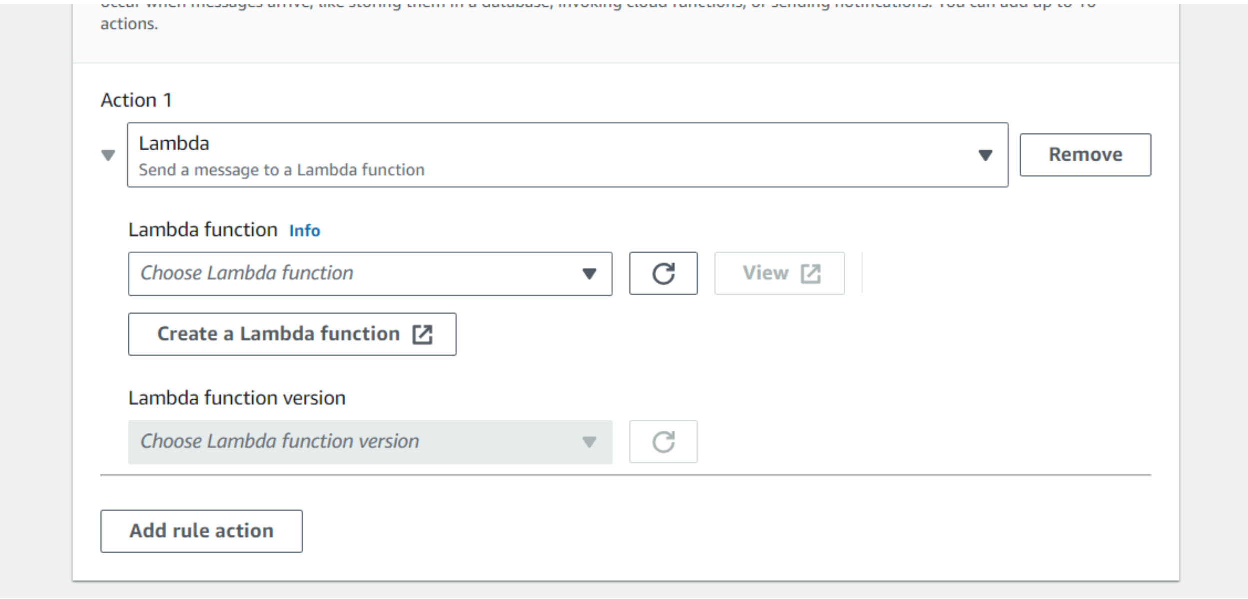
2016-03-23 ▼

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.

```
SELECT * FROM 'esp32/pub'
```

- Finally, we specify the Gateway Initialization Lambda function, which will complete the initialization process.



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

Lambda
Send a message to a Lambda function

Lambda function Info

Choose Lambda function

Lambda function version

Choose Lambda function version

After the rule is created, we can create our APIs to get sensor information, which will be used by the Elastic Search and Control Dashboard.

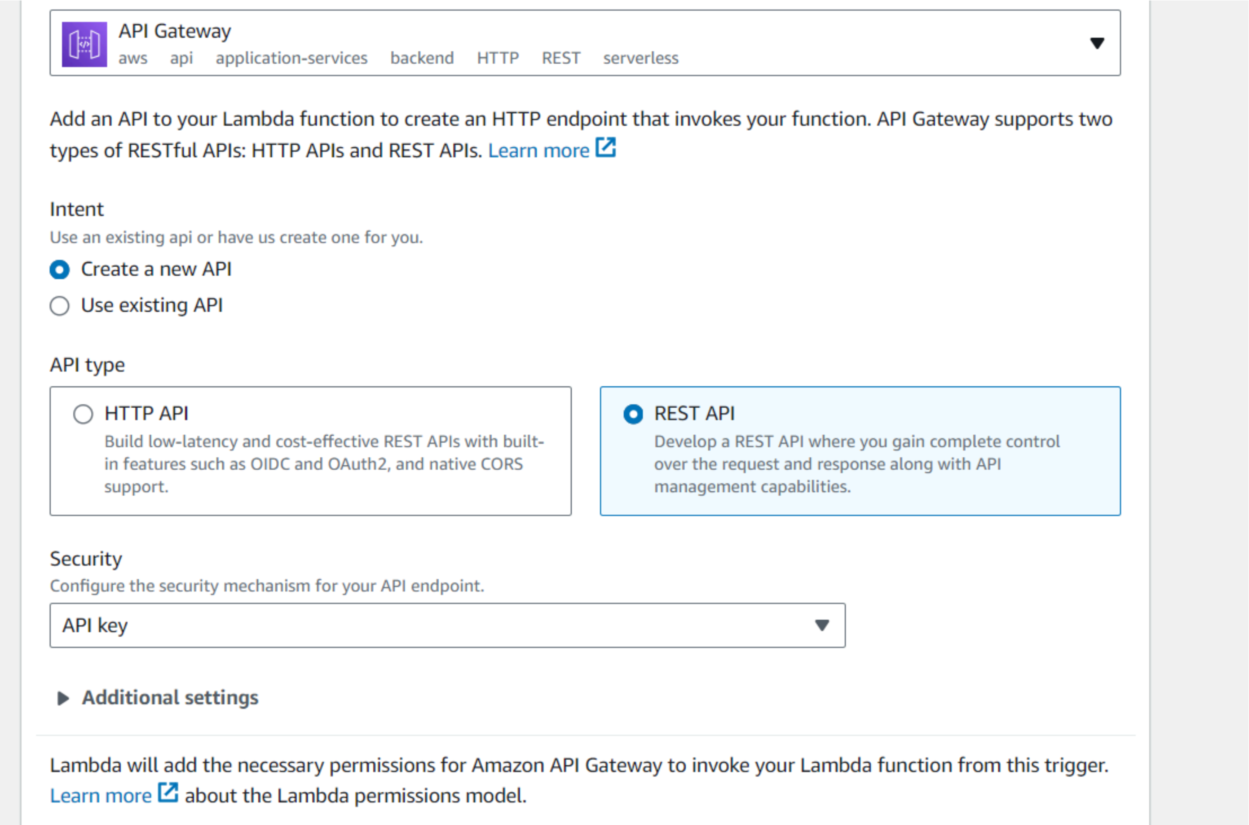
Step 4: Adding API Triggers for Sensor Data Lambda Function and Control Lambda Function

To enable API triggers for our Sensor Data Lambda Function and Control Lambda Function, follow these steps:

1. Sensor Data API Handler Lambda Function

- Add a trigger in the form of a REST API endpoint.

- Configure the REST API using the settings provided in the picture.



API Gateway
aws api application-services backend HTTP REST serverless ▼

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. Learn more

Intent
Use an existing api or have us create one for you.

● Create a new API
○ Use existing API

API type

○ HTTP API
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

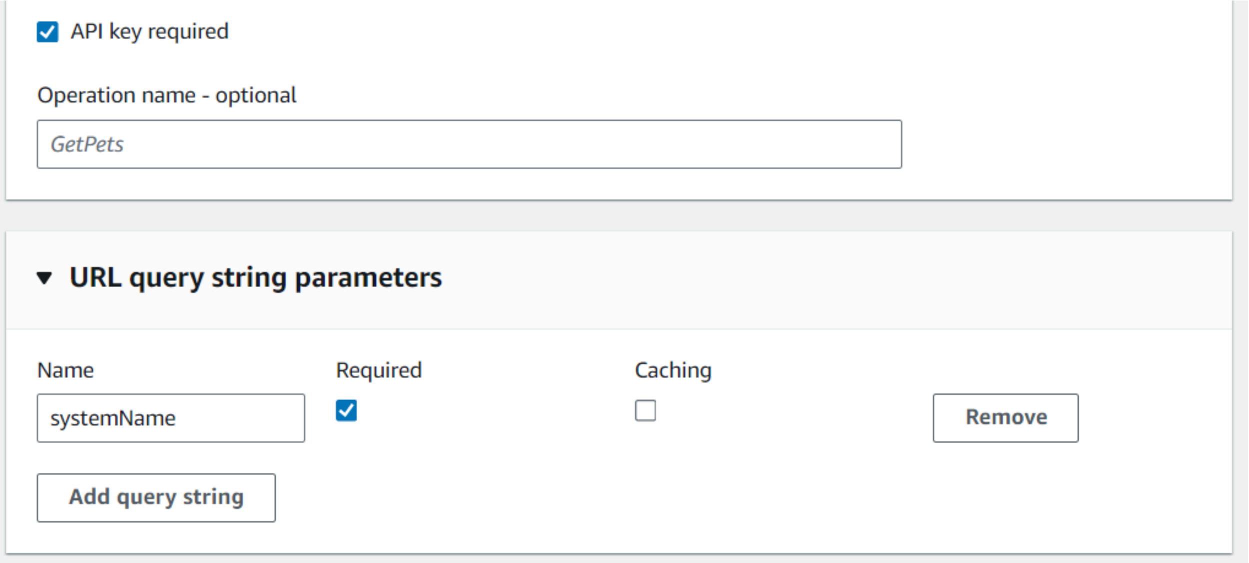
● REST API
Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Security
Configure the security mechanism for your API endpoint.
API key ▼

▶ Additional settings

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

- Navigate to the AWS API Gateway console to manage the API.
- Apply the settings specified in the picture to the method associated with the Sensor Data API Handler Lambda Function.



☑ API key required

Operation name - optional
GetPets

▼ **URL query string parameters**

Name	Required	Caching	
systemName	☑	□	Remove

Add query string

2. Control Lambda Function:

- Add a trigger using the same method as the Sensor Data Lambda Function.
- Keep the default configuration for this trigger without making any changes.

By setting up API triggers in this manner, we establish endpoints through which external systems can interact with our Lambda functions. This enables seamless integration of our IoT infrastructure with external applications or services like our Elastic Search Dashboard and Control Dashboard.