



Getting Started with Kerlink WAL-e and Amazon Web Services

Getting Started Guide

Getting Started with Kerlink WAL-e and Amazon Web Services

	Redaction	Validation	Approbation
Trigram	JCA	GBO	YDE
Date	2020-11-13		
Signature			

Version	Edits
1.0	Initial version
1.1	Update based on AWS feedback
1.2	Lambda code improvement
1.3	Typo corrections, clarifications add references

Reference	Description
[1]	https://wikikerlink.fr/wirnet-productline/doku.php?id=wiki:lora:aws

Table of content

1 Prerequisites.....	4
1.1 What you need	4
1.2 Application overview	4
2 Application Design.....	5
2.1 AWS IoT Core.....	5
2.1.1 Create rule.....	5
2.1.2 Add an action	7
2.1.3 Configure action	9
2.1.4 Finalize rule	10
2.2 AWS Lambda	12
2.2.1 Function code	13
2.2.2 Deploy	14
2.3 AWS Simple Notification Service.....	15
2.3.1 Create topic	16
2.3.2 Create subscription	18
2.4 AWS Lambda permissions setup	20
2.4.1 Lambda execution role.....	21
2.4.2 Attach a new policy	21
2.4.3 Attach policy to lambda role	25
2.5 Follow application execution	27
2.5.1 AWS IoT Core test.....	27
2.5.2 AWS Cloudwatch	27
2.5.3 AWS SNS publish	27
3 Going further	27

Table of figures

Figure 1: WAL-e application on AWS overview.....	5
Figure 2: AWS IoT Core – Rules	6
Figure 3: AWS IoT Core - Create a rule.....	7
Figure 4: AWS IoT Core – Select an action	8
Figure 5: AWS IoT Core - Configure Lambda action	9
Figure 6: AWS lambda -- Create function.....	10
Figure 7: AWS IoT Core rule - Create rule	11

Figure 8: AWS IoT Core - Successfully created rule.....	12
Figure 9: AWS Lambda - empty function	13
Figure 10: AWS Lambda – WAL-e decoder Python code	14
Figure 11: AWS Lambda – Deploy	15
Figure 12: AWS SNS - landing page	16
Figure 13: AWS SNS - Create topic	17
Figure 14: AWS SNS - topic created successfully	18
Figure 15: AWS SNS - Create subscription	19
Figure 16: AWS SNS - Subscription created	20
Figure 17: AWS SNS - Subscription confirmed	20
Figure 18: AWS Lambda – Permissions	21
Figure 19: AWS IAM - Role summary	22
Figure 20: AWS IAM - Add permissions.....	23
Figure 21: AWS IAM - Create policy	24
Figure 22: AWS IAM - review policy	25
Figure 23: AWS IAM –Policy usage.....	26
Figure 24: AWS IAM - Attach policy	26

This guide will walk you through the creation of a cloud application for Kerlink WAL-e LoRaWAN sensor on Amazon Web Services (AWS).

1 Prerequisites

This tutorial is built on top of preceding Getting Started guides where you should have enabled your Kerlink WAL-e and routed its messages onto AWS IoT Core. Please refer to the following documents at [1]:

- Getting Started with Kerlink gateways AWS IoT Core for LoRaWAN
- Getting Started with Kerlink WAL-e AWS IoT Core for LoRaWAN

1.1 What you need

To follow this guide, you need a Kerlink WAL-e configured and connected to AWS IoT Core. You also need an AWS account with sufficient permissions to use the following services

- AWS IoT Core
- AWS Lambda
- AWS Simple Notification Service

NOTE – all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions such as publishing messages or subscribing to topics with specific scope and context. The specific permission policies can vary for your use cases. Identify the permission policies that best meet your business and security requirements.

For sample policies, refer to

<https://docs.aws.amazon.com/iot/latest/developerguide/example-iot-policies.html> . Also refer to <https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html>

1.2 Application overview

This guide will walk you through the process of creating an application which triggers a notification, either by email or by SMS, when the Kerlink WAL-e LoRaWAN sensor is activated. WAL-e has two modes of operation - It triggers an alarm either when the button is pressed or when a movement is detected.

No initial knowledge is required on LoRaWAN, nor on AWS services. It is recommended though to learn about AWS Policies and Permissions, as well as AWS pricing to avoid misuse of the services. All actions in this tutorial are covered by AWS free tier.

2 Application Design

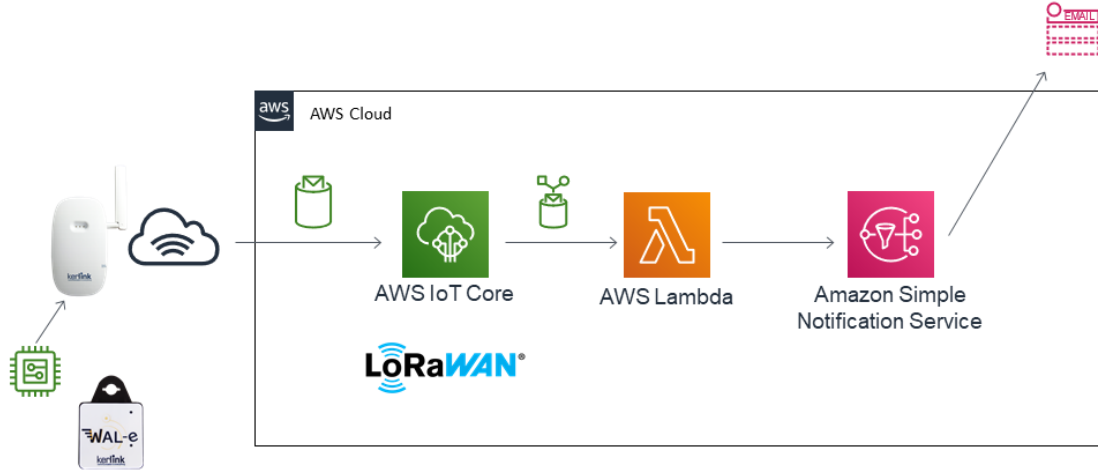


Figure 1: WAL-e application on AWS overview

The application we are about to design relies on three AWS services, listed in the figure above. Messages are coming from the LoRaWAN Network into AWS IoT Core for LoRaWAN. We will define a rule to trigger an AWS Lambda function. That function will decode the payload sent by the sensor and trigger a notification on AWS SNS. Finally, SNS will alert the user, forwarding the notification by email or by SMS.

2.1 AWS IoT Core

2.1.1 Create rule

First, create a rule in AWS IoT Core. Go to AWS IoT Core console at console.aws.amazon.com/iot, select “Act” >> “Rules”, and click the “Create” button.

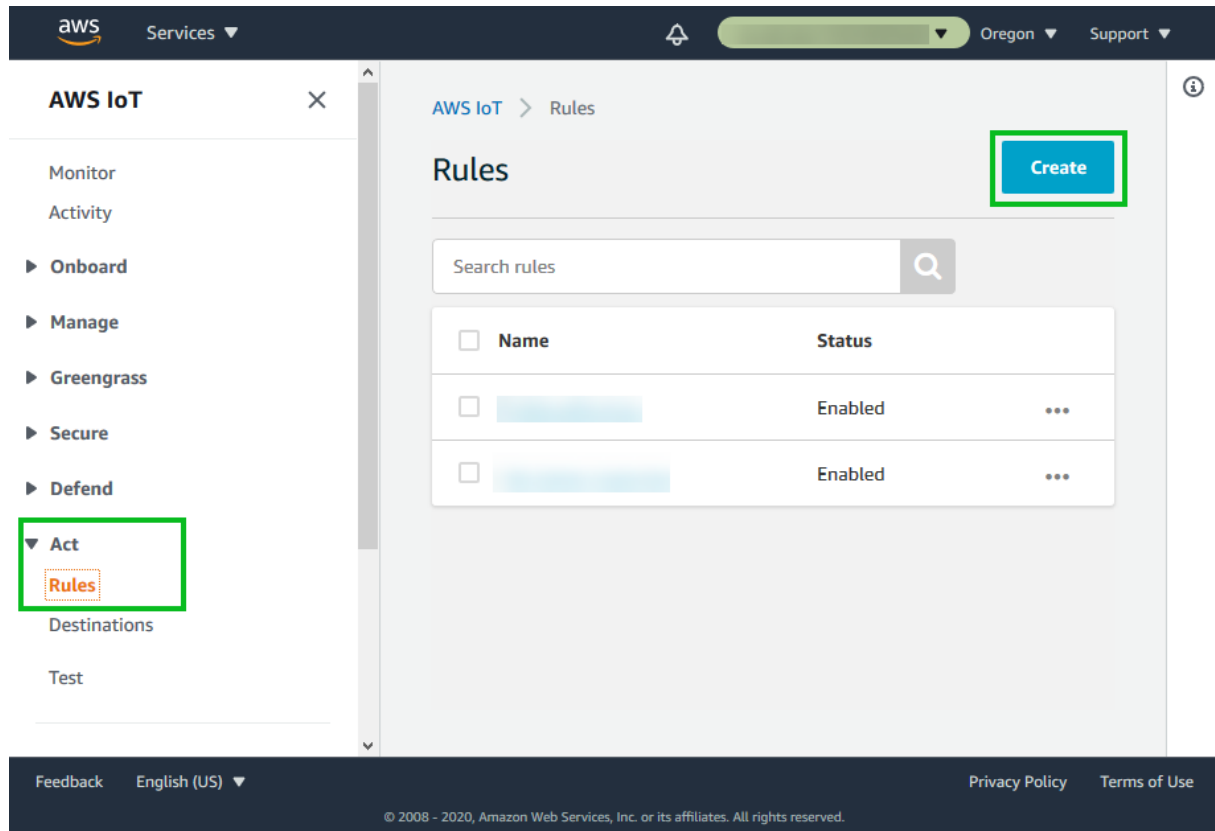
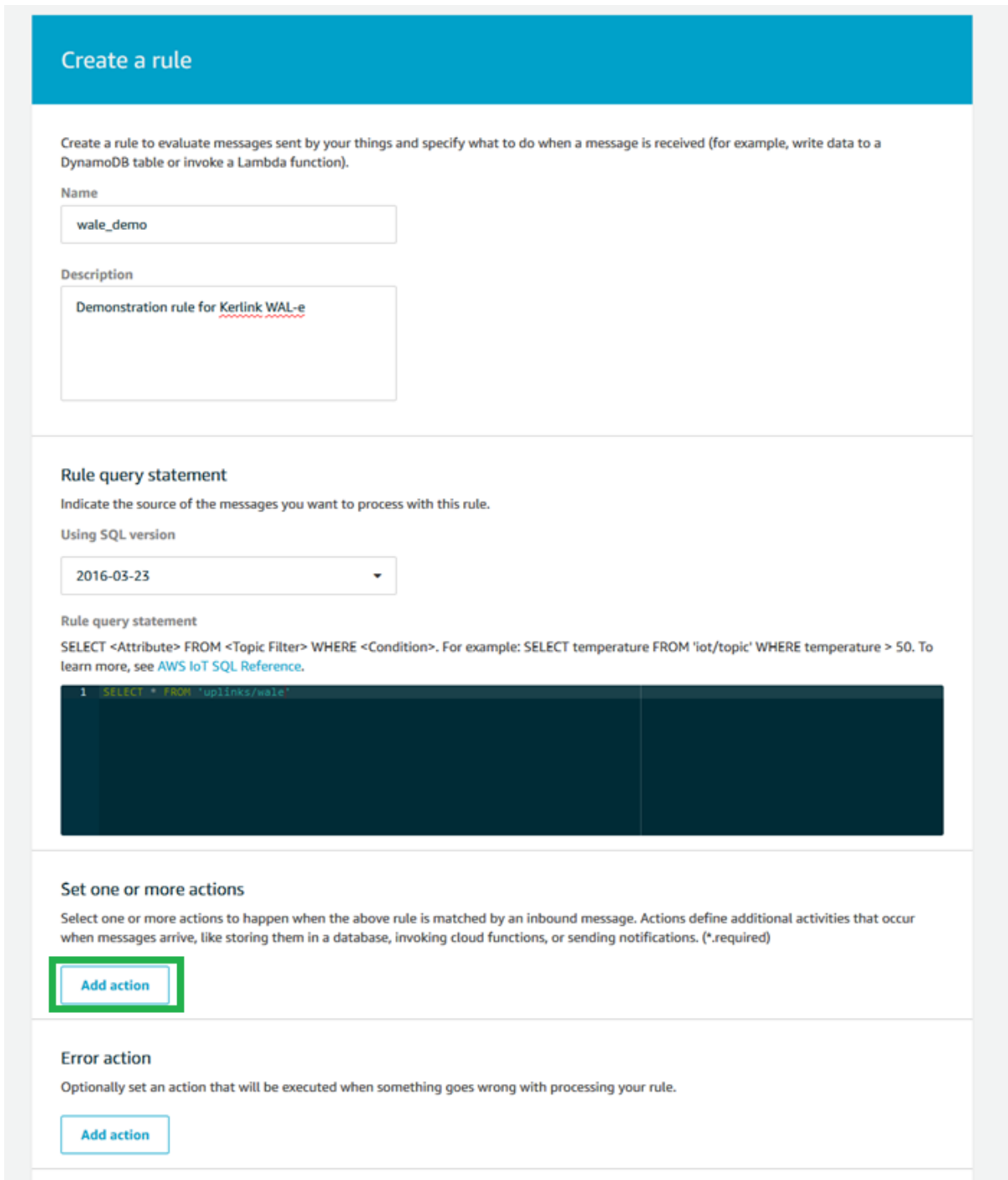


Figure 2: AWS IoT Core – Rules

Give the rule a name and a description. Then, write the Rule query statement as follows:

```
SELECT * FROM 'uplinks/wale'
```

Where `uplinks/wale` is the AWS IoT Core topic the messages are published to. This will select all the messages from that topic to run this rule.



Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name
wale_demo

Description
Demonstration rule for Kerlink WAL-e

Rule query statement
Indicate the source of the messages you want to process with this rule.

Using SQL version
2016-03-23

Rule query statement
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 'uplinks/wale'
```

Set one or more 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. (*.required)

Add action

Error action
Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

Figure 3: AWS IoT Core - Create a rule

Next step is to add an action in that rule. Click “Add action” button.

2.1.2 Add an action

Select the “Send a message to a Lambda function” action in the list.

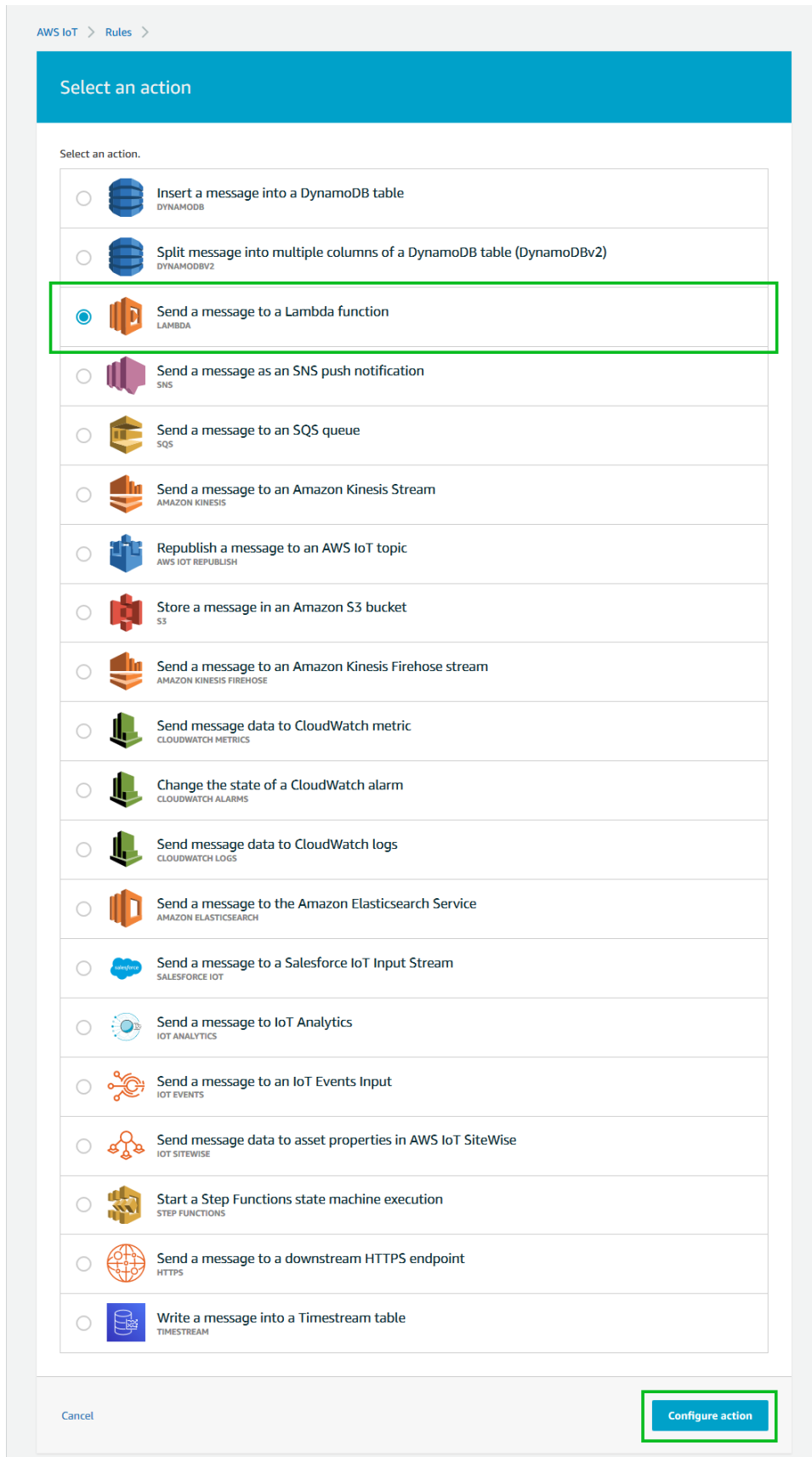


Figure 4: AWS IoT Core – Select an action

2.1.3 Configure action

Continue with clicking the “Configure Action” button at the bottom of the page.

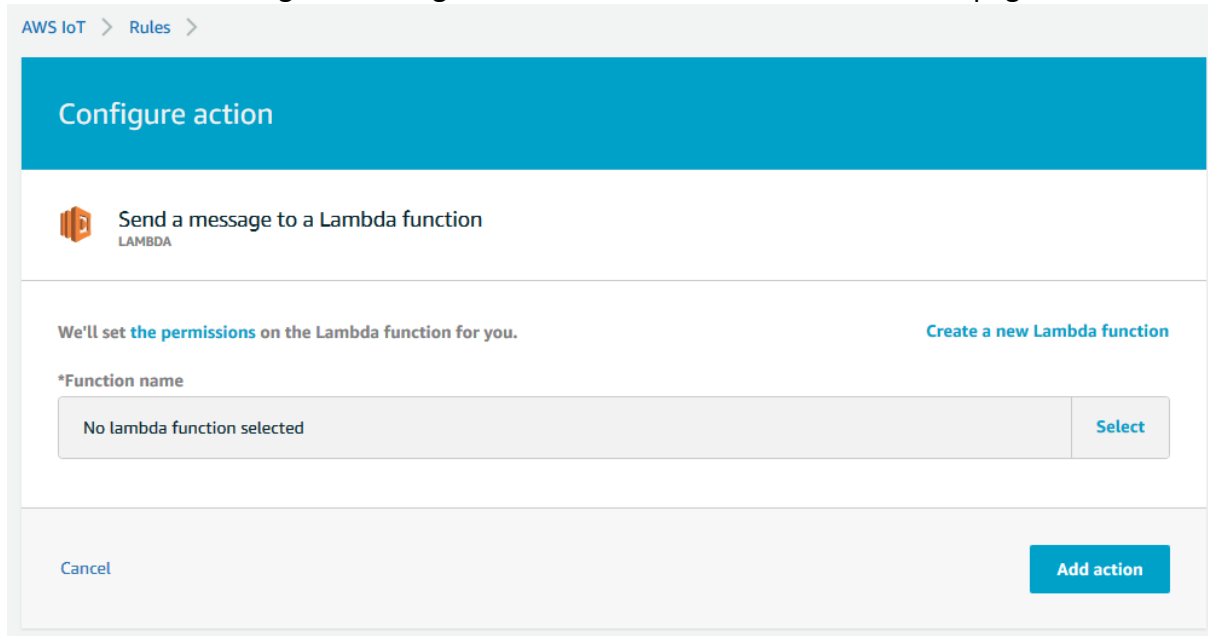


Figure 5: AWS IoT Core - Configure Lambda action

You need to create a lambda function for that action. Select “Create a new Lambda function” link. AWS interface will redirect you to the AWS Lambda page. Refer to Section 2.2 to configure your function.

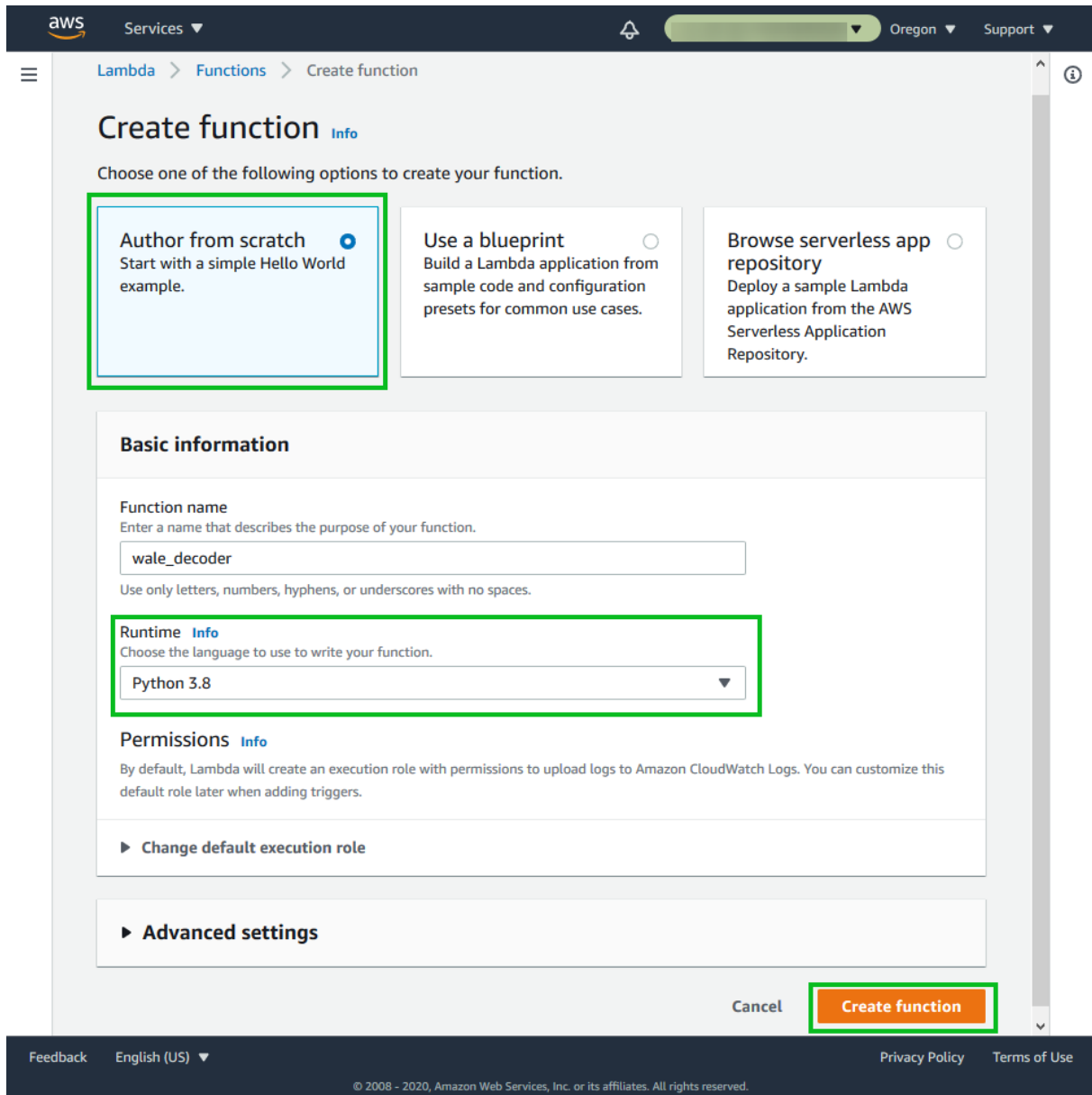


Figure 6: AWS lambda -- Create function

For the moment, simply create a Python 3.8 function from scratch.


When the lambda is created, you can return to the AWS IoT Core Rule “Configure Action” (Figure 5) and click “Select” and eventually “Refresh” to select your freshly created function. Finish this procedure clicking “Add action” button.

2.1.4 Finalize rule

Finally, once your action is added to the rule, create the rule on AWS IoT Core using the “Create Rule” button at the bottom of the page.

Set one or more 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. (*.required)

 **Send a message to a Lambda function** Remove Edit ▶
wale_decoder

[Add action](#)

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

[Add action](#)

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name Value Clear

[Add another](#)

[Cancel](#) **Create rule**

Figure 7: AWS IoT Core rule - Create rule

The rule is now created. You can now switch back to AWS Lambda interface to write the decoding function.

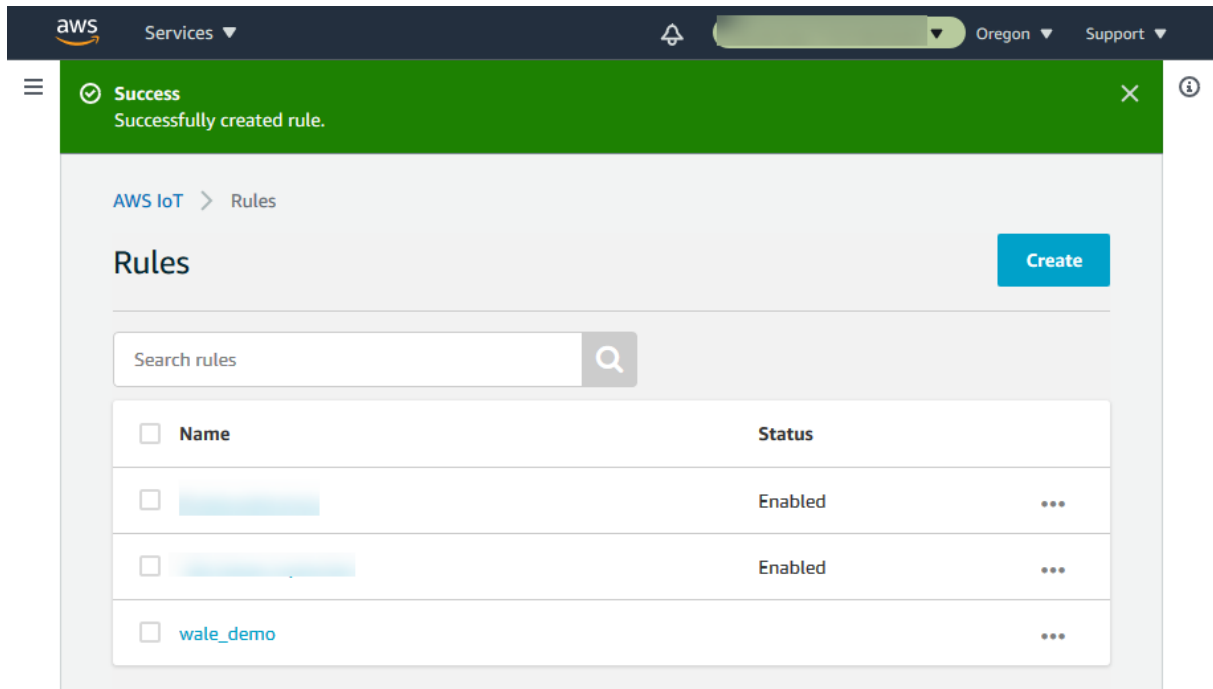


Figure 8: AWS IoT Core - Successfully created rule

2.2 AWS Lambda

Browse back to AWS Lambda page to configure your function. Either go to the browser tab you left while configuring AWS IoT Core action or go to AWS console and select Lambda service and select your function.

Your AWS Lambda function should look like the figure below. Note the link with AWS IoT in the trigger part of the Designer graph.

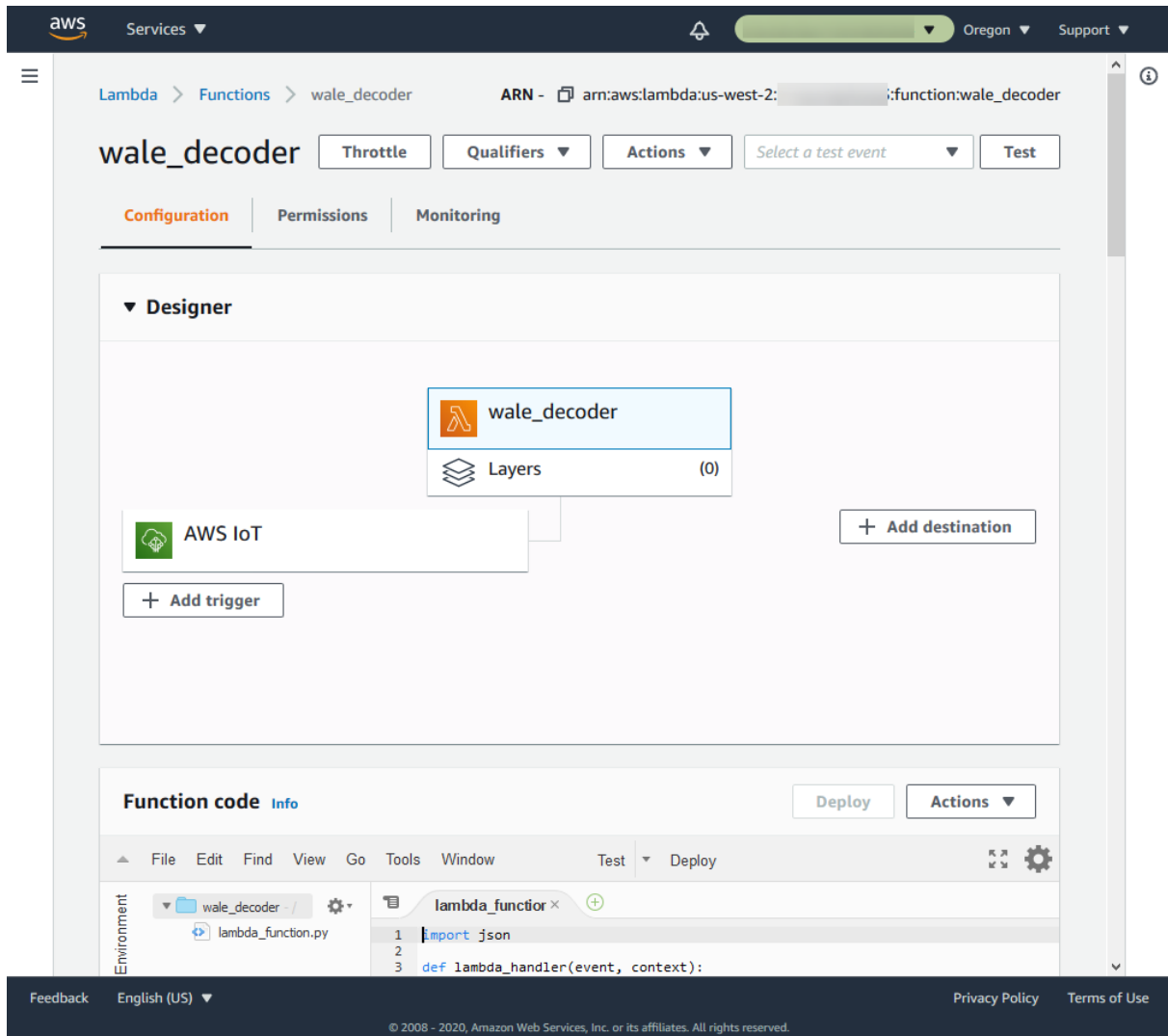


Figure 9: AWS Lambda - empty function

2.2.1 Function code

Browse to the Function code part of the page. There is a code editor to write the instructions of your function, with a template of Python code.

Copy and paste the following code snippet to the `lambda_function.py` code area.

```
import json
import base64
import struct
import boto3

def lambda_handler(event, context):
    # FPort = 2 is required
    if event['ApplicationId'] != 2:
        return
```

```
data = base64.b64decode(event['PayloadData'])
values = struct.unpack_from("<hhhhh", data)
keys = ('version', 'event', 'temp', 'humidity',
        'voltage')

body = dict(zip(keys, values))
body.update(event)

client = boto3.client('sns')
response = client.publish(
    TopicArn = 'arn:aws:sns:us-west-
2:000000000000:IoT-to-email',
    Message = "Motion detected on
{}".format(body['Metadata']['LoRaWAN']['DevEUI'][8:]),
    Subject = "Motion detected"
)
```

Figure 10: AWS Lambda – WAL-e decoder Python code

This code is hooked to an event, triggered by the previously defined AWS IoT Core rule. The `event` argument is a Python dictionary from the JSON received from AWS IoT Core. The code first filters on LoRaWAN FPort 2, then decodes the base64 payload data and unpacks each value into a Python dictionary named `body`. The event dictionary keys are based on AWS IoT Core for LoRaWAN service. Depending on the LoRaWAN Network Server used, those keys may be different and should be adapted.

We then use AWS `boto3` Python module to publish the decoded elements to an AWS SNS topic named `IoT-to-email`, adding an email subject and an email body containing the DevEUI of the device which triggered the lambda.

Note that you need to change the AWS SNS topic ARN with your own. See Section 2.3.1 to create that topic.

Note that you will need to add permission to your lambda function to publish on that topic. Refer to Section 2.4 to add the permission required.

To use SMS instead of email notification, as well as other functionalities of the `boto3` module, please refer to the [boto3 documentation](#).

2.2.2 Deploy

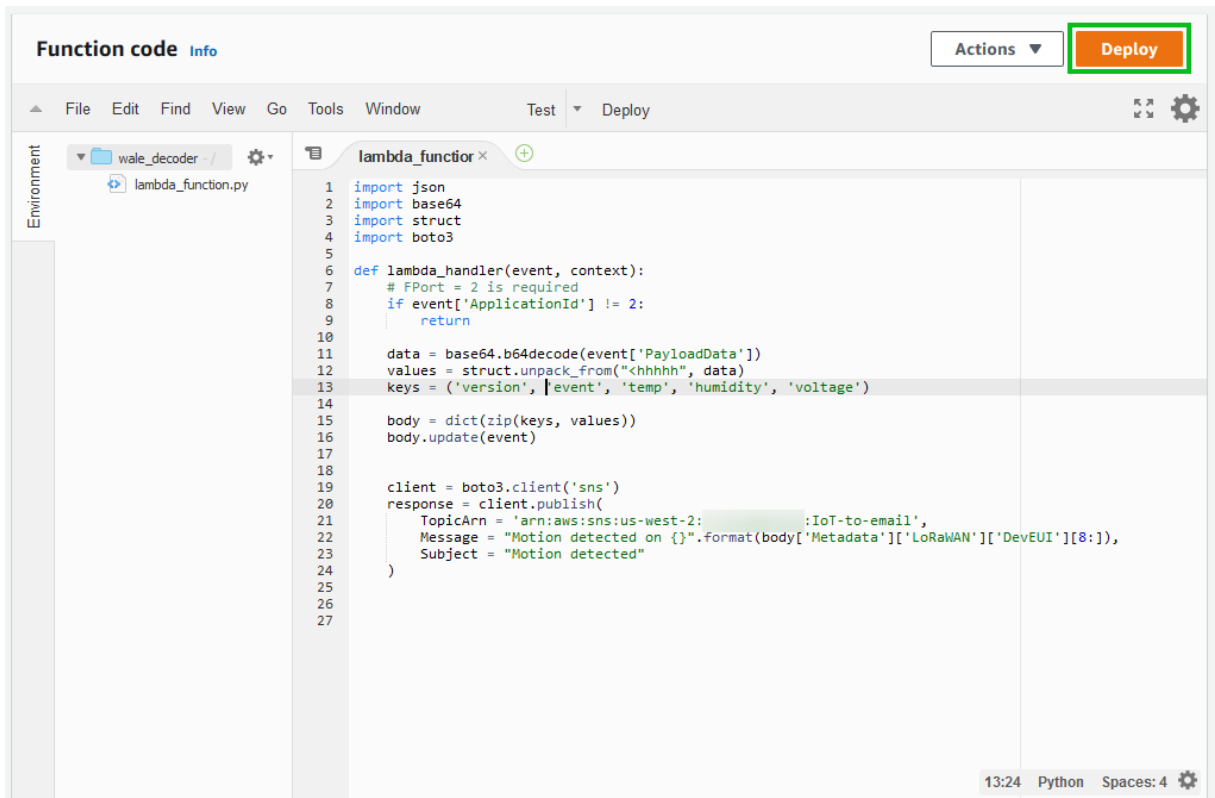


Figure 11: AWS Lambda – Deploy

Hit the “Deploy” button to enable the function.

The code is ready. The next step is to configure AWS Simple Notification Service to receive an email when an uplink occurs.

2.3 AWS Simple Notification Service

Go back to the AWS console home and look for Simple Notification Service. The service starts with a landing page.

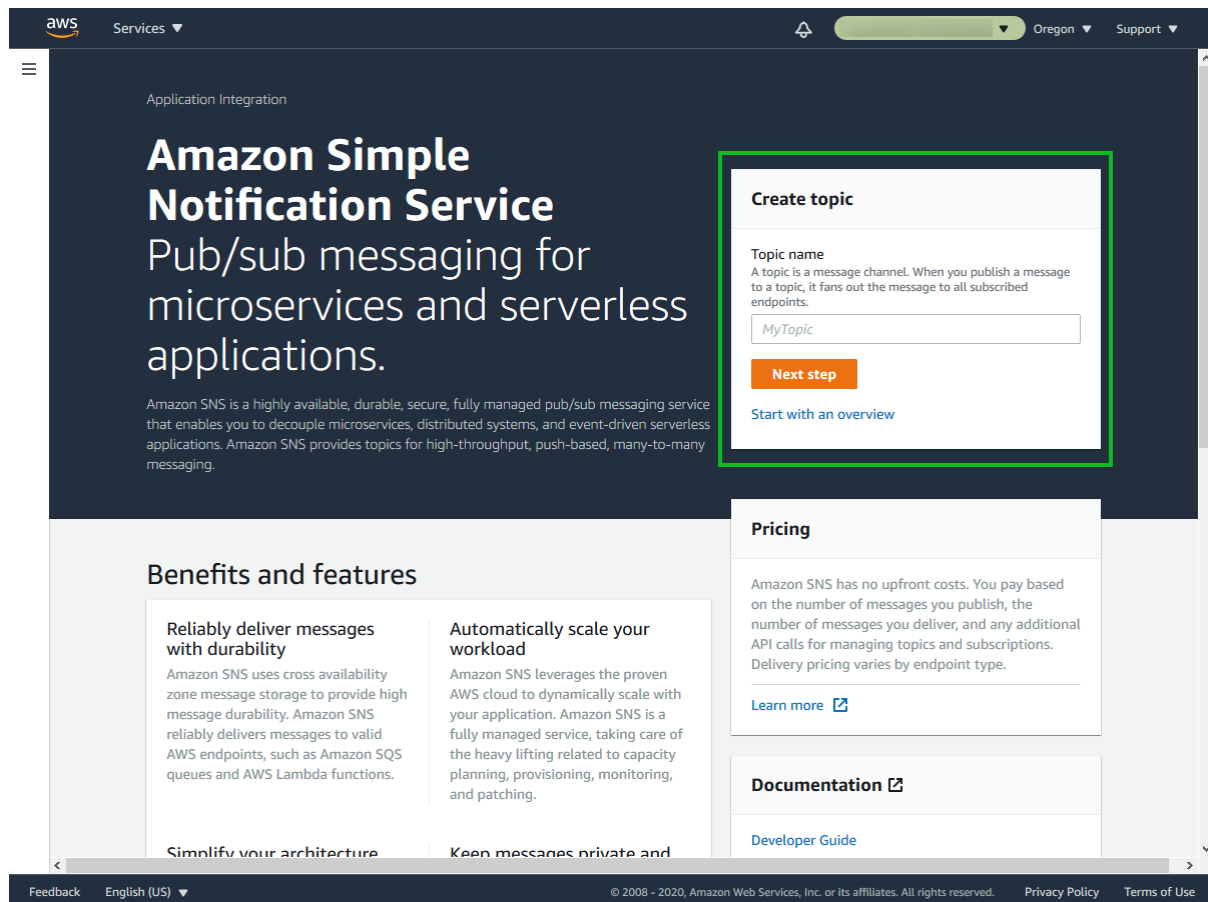


Figure 12: AWS SNS - landing page

Type in the topic we chose in the AWS Lambda code and click “Next step”.

2.3.1 Create topic

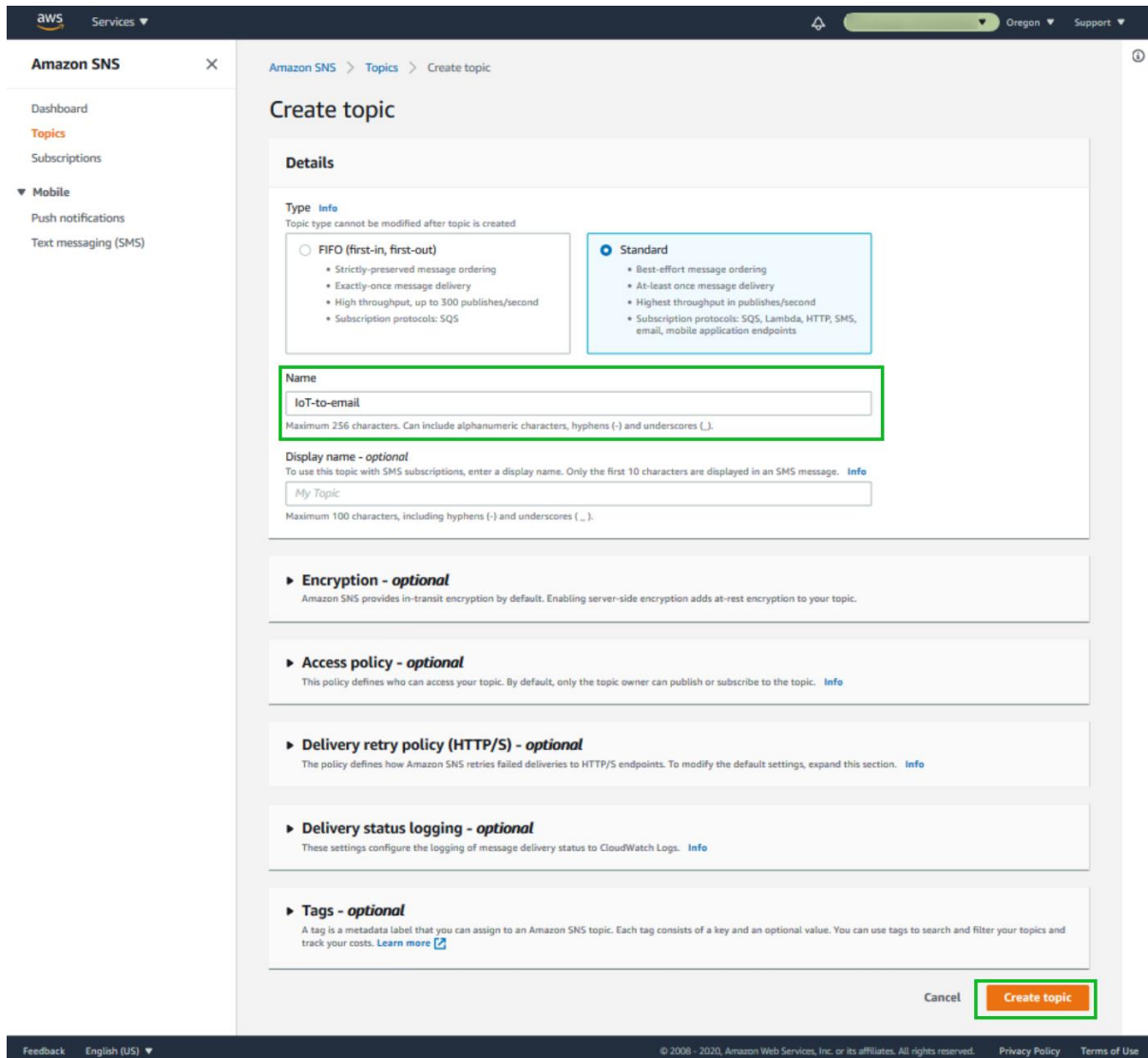


Figure 13: AWS SNS - Create topic

Simply check the name and hit “Create topic” button at the bottom of the page. Refer to AWS SNS documentation to know more about the other options.

Once the topic created, note the ARN for your lambda function (Refer to Section 2.2.1).

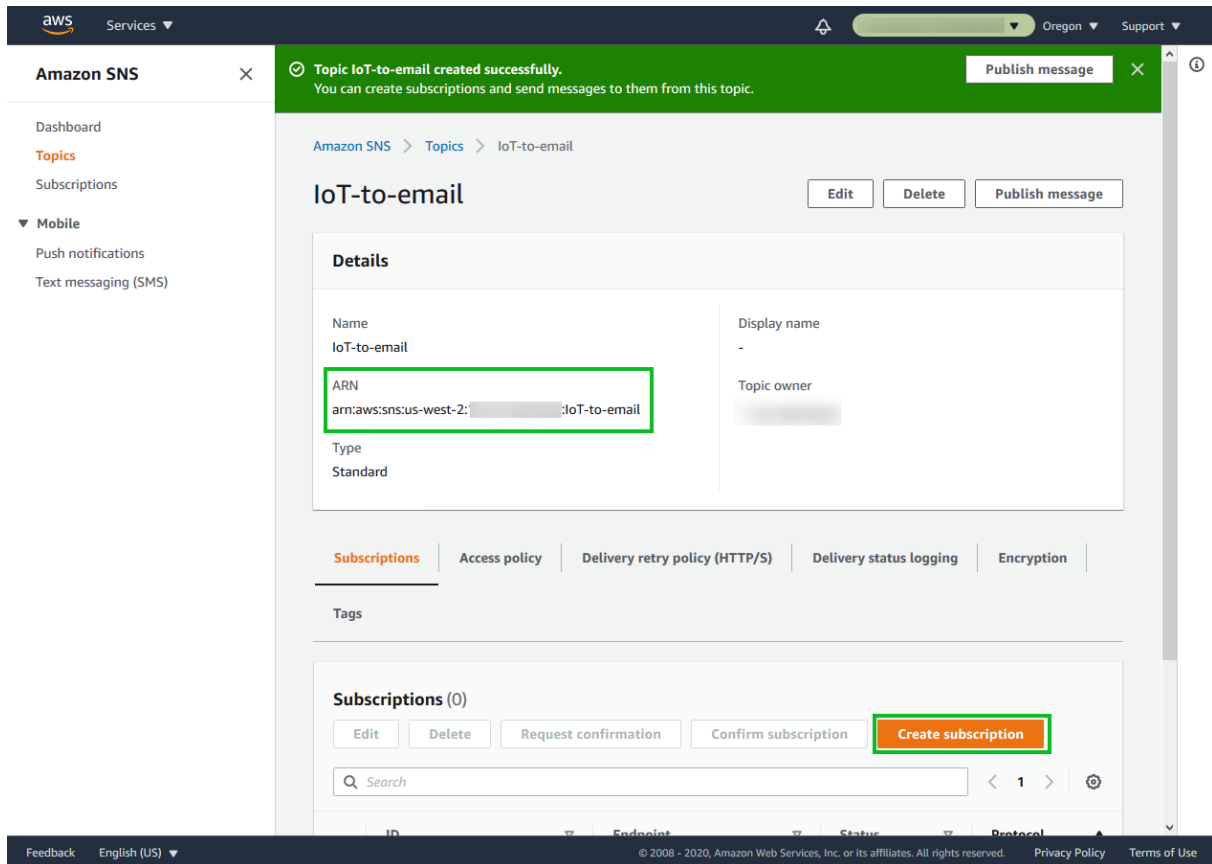


Figure 14: AWS SNS - topic created successfully

2.3.2 Create subscription

On the topic page, there is a subscription section. Hit the “Create subscription”. This will setup the email notification for the topic we just created.

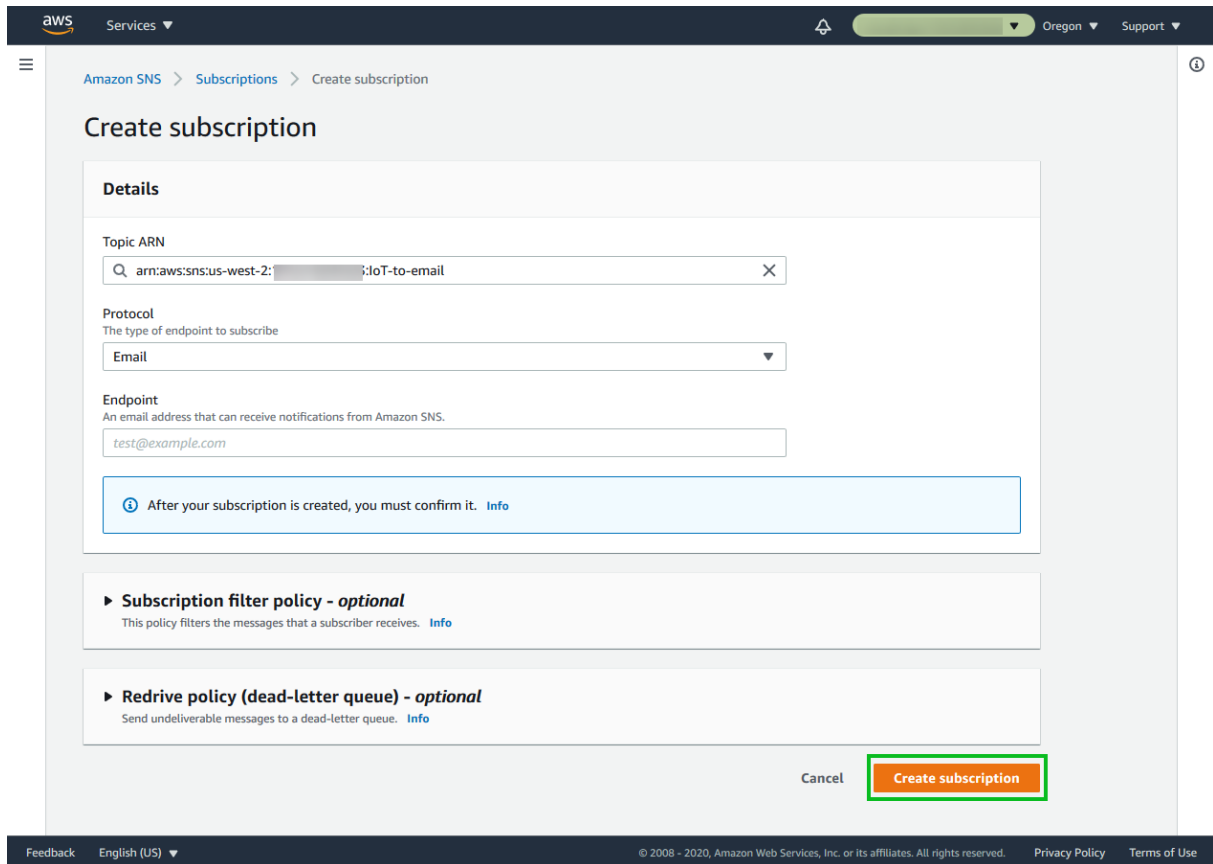


Figure 15: AWS SNS - Create subscription

Choose “Email” under the Protocol list and enter the e-mail of the notification recipient in the Endpoint field.

You can also select “SMS” if you prefer this notification method and add the phone number to be notified. Service charges may apply.

Finally, hit “Create subscription” button.

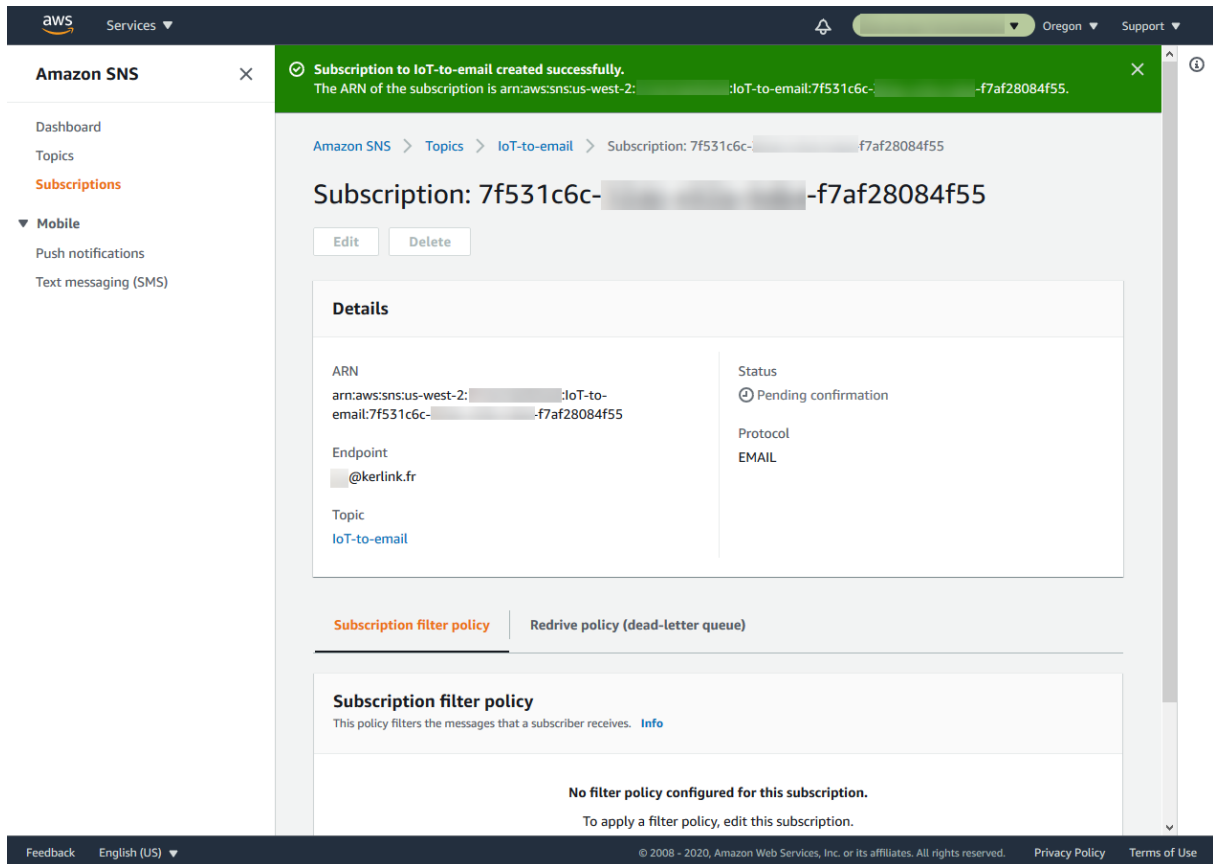


Figure 16: AWS SNS - Subscription created

Note that the subscribed email needs to be confirmed. You will receive a confirmation email immediately after creating the subscription. Click on the "Confirm subscription" link.



Simple Notification Service



Figure 17: AWS SNS - Subscription confirmed

2.4 AWS Lambda permissions setup

The final step is to allow the lambda function we created to post to the SNS topic.

2.4.1 Lambda execution role

To add permission to the lambda function, go to AWS Lambda, and select the function. Switch to the “Permissions” tab of the function and click on the role name under “Execution role”.

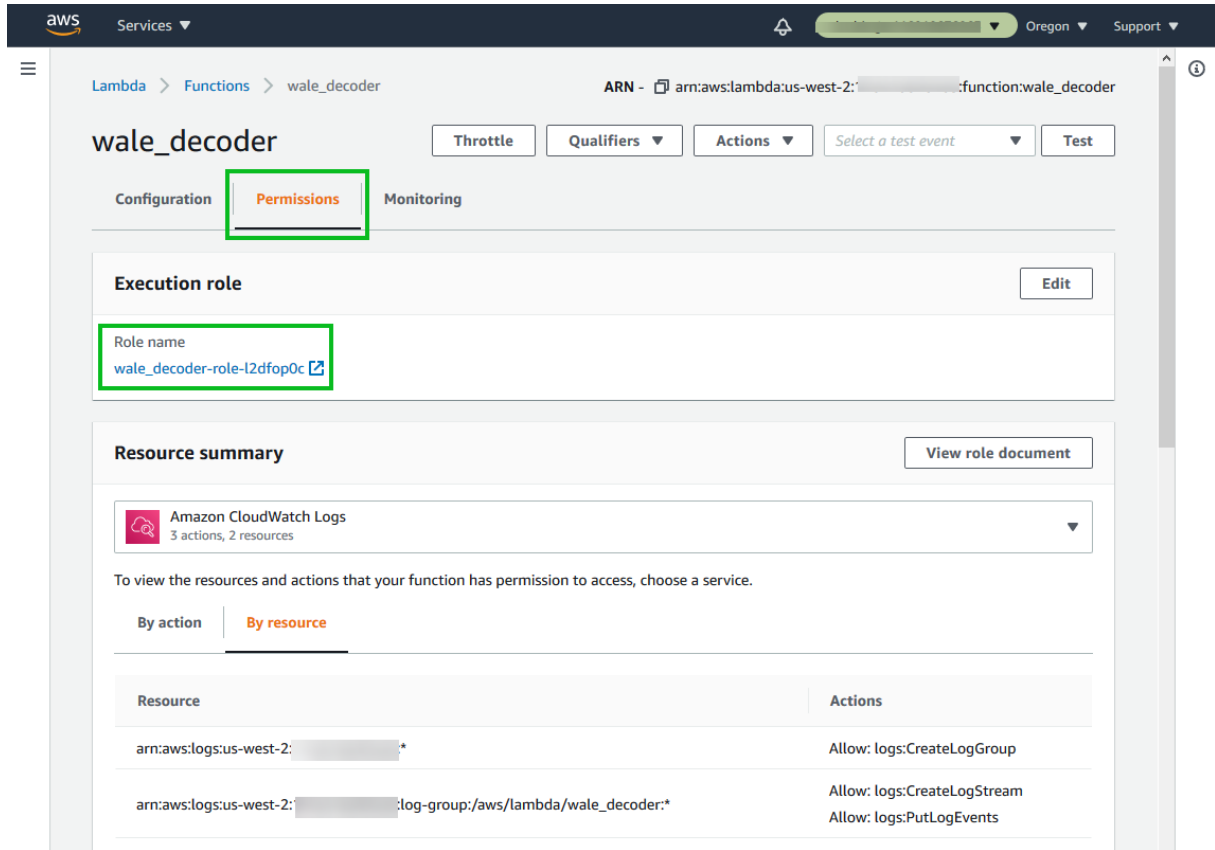


Figure 18: AWS Lambda – Permissions

This link leads to AWS Identity and Access Management (IAM), on the role summary of the function.

2.4.2 Attach a new policy

The screenshot shows the AWS IAM console interface. On the left is a navigation menu for Identity and Access Management (IAM). The main content area displays the 'Summary' tab for the role 'wale_decoder-role-l2dfop0c'. The summary includes details such as Role ARN, Role description (Edit), Instance Profile ARNs, Path (/service-role/), Creation time (2020-11-13 17:24 UTC+0100), Last activity (2020-11-13 21:07 UTC+0100 (Today)), and Maximum session duration (1 hour Edit). Below the summary are tabs for Permissions, Trust relationships, Tags, Access Advisor, and Revoke sessions. The 'Permissions' tab is active, showing 'Permissions policies (1 policy applied)'. A blue 'Attach policies' button is highlighted with a green box. Below it is a table with one row: 'AWSLambdaBasicExecutionRole-7eb4...' with a 'Managed policy' type. There is also a '+ Add inline policy' button and a 'Permissions boundary (not set)' section.

Policy name	Policy type
AWSLambdaBasicExecutionRole-7eb4...	Managed policy

Figure 19: AWS IAM - Role summary

Click “Attach policies” button under the Permissions tab.

aws Services Global Support

Add permissions to wale_decoder-role-l2dfop0c

Attach Permissions

Create policy

Filter policies Search Showing 620 results

	Policy name	Type	Used as
<input type="checkbox"/>	AdministratorAccess	Job function	Permissions policy (1)
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessLifsizeDelegatedAccessPolicy	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	AWS managed	None
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	AWS managed	None
<input type="checkbox"/>	AmazonAPIGatewayInvokeFullAccess	AWS managed	None
<input type="checkbox"/>	AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	None
<input type="checkbox"/>	AmazonAppFlowFullAccess	AWS managed	None
<input type="checkbox"/>	AmazonAppFlowReadOnlyAccess	AWS managed	None
<input type="checkbox"/>	AmazonAppStreamFullAccess	AWS managed	None
<input type="checkbox"/>	AmazonAppStreamReadOnlyAccess	AWS managed	None

Cancel Attach policy

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Figure 20: AWS IAM - Add permissions

On the next page, click “Create policy” button to add a new policy for that role.

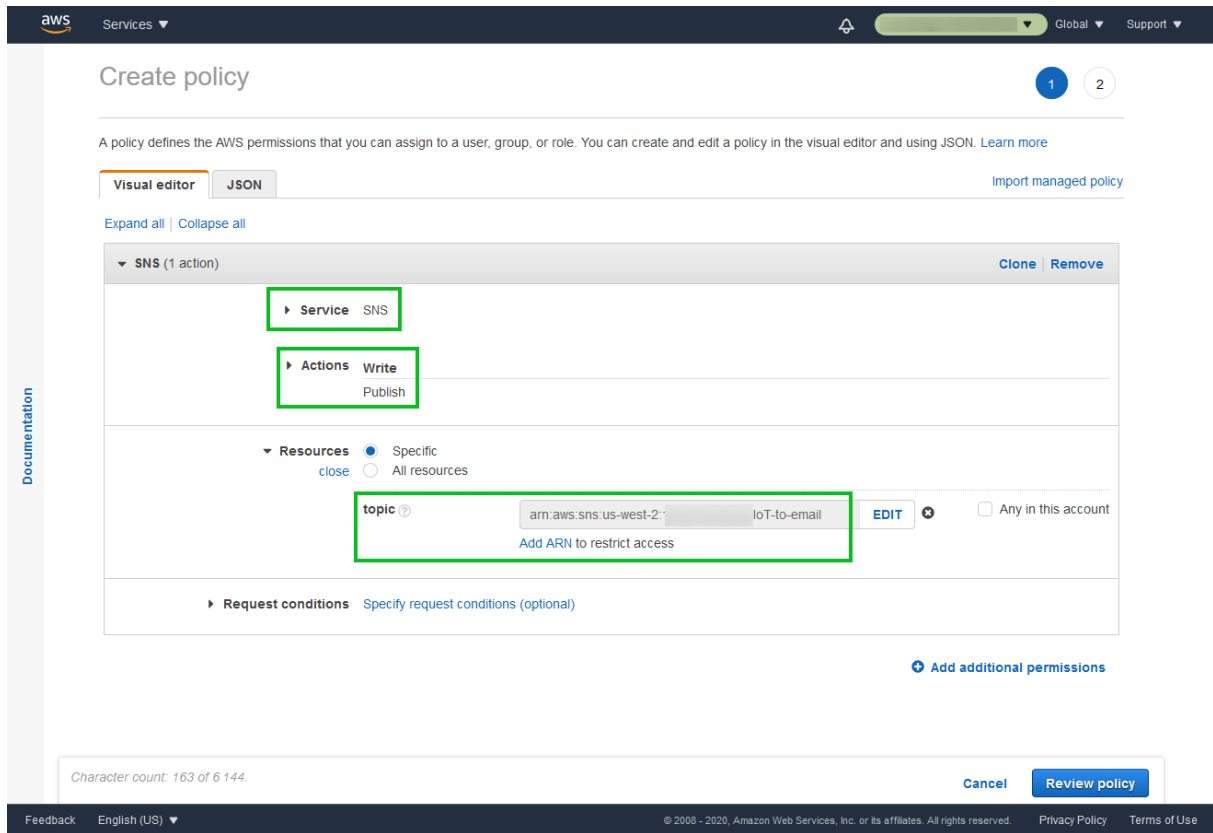


Figure 21: AWS IAM - Create policy

Select the “SNS” service. On the actions list, tick “Publish” under “Write” actions. Finally, select a specific resource, giving the topic ARN you just created in AWS SNS. Hit the “Review policy” button on the bottom of the page.

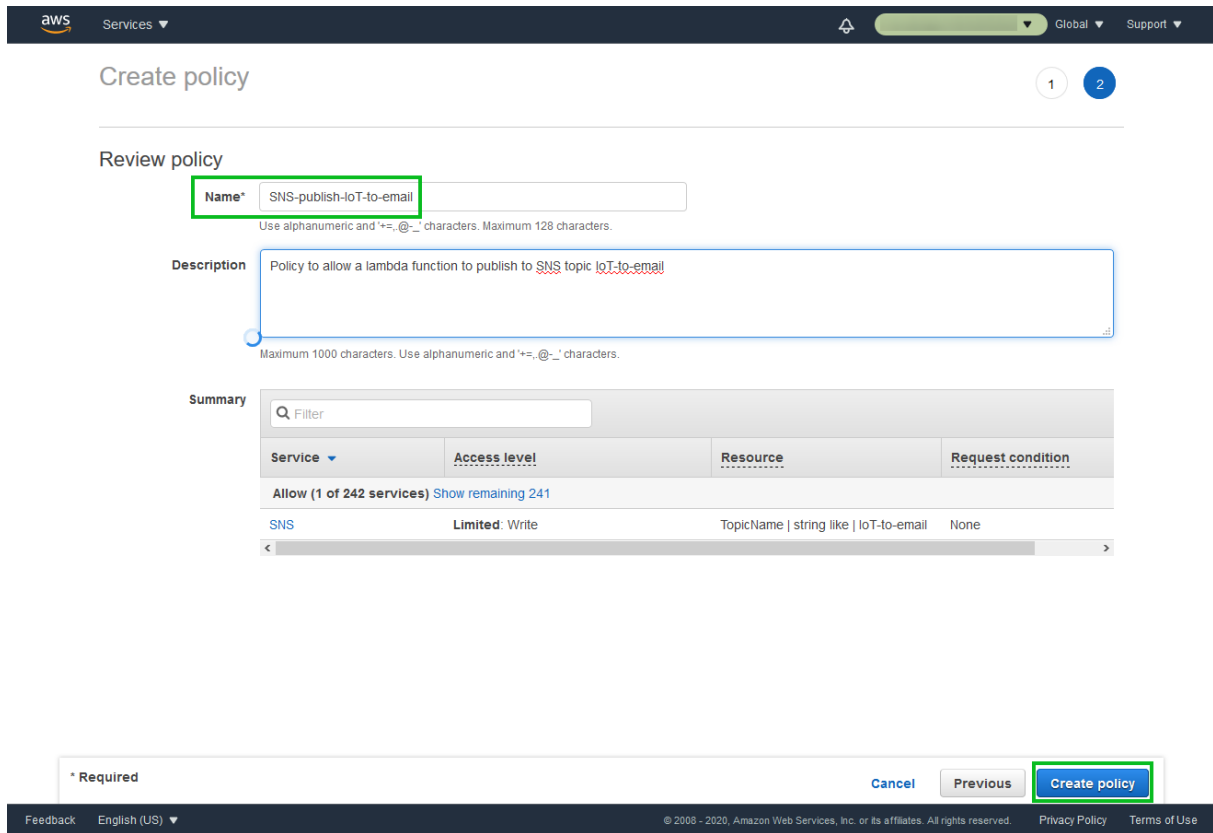


Figure 22: AWS IAM - review policy

Give the policy a name and a description. Finally, click the “Create policy” button.

2.4.3 Attach policy to lambda role

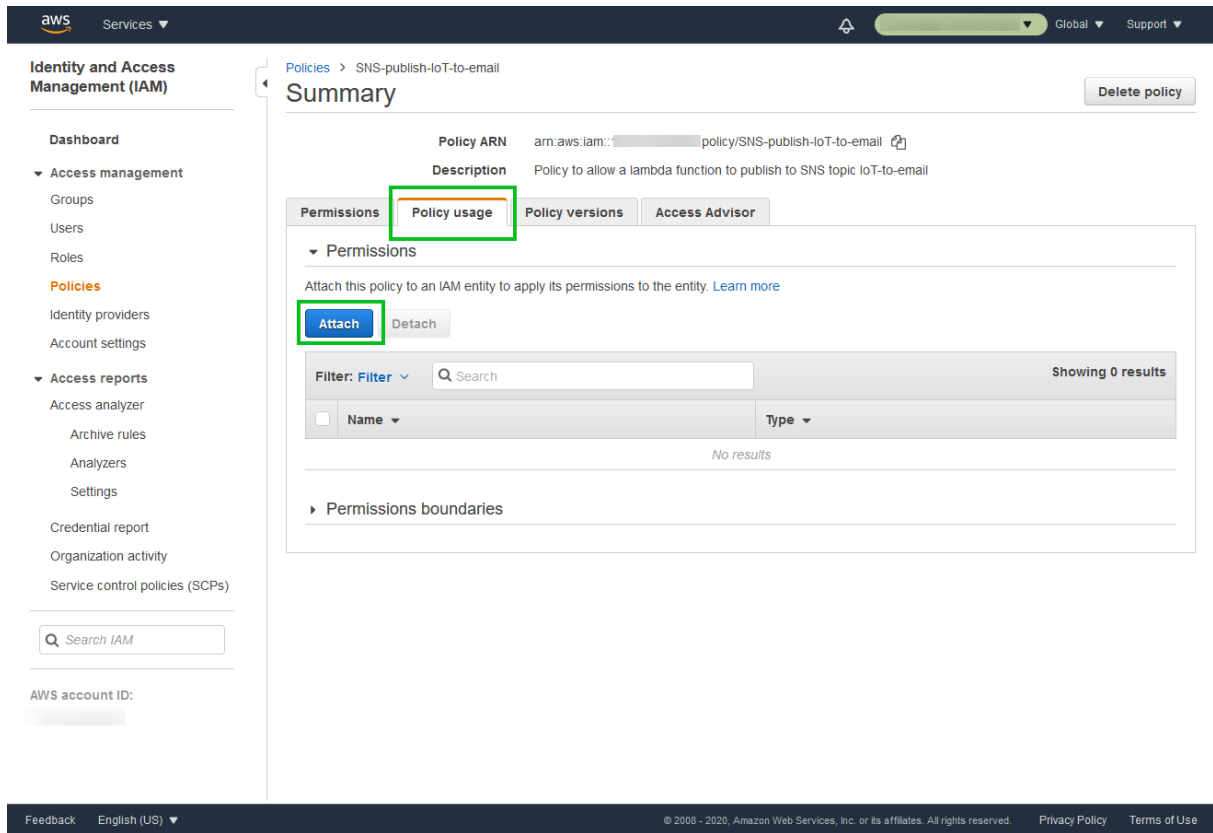


Figure 23: AWS IAM –Policy usage

Navigate to your newly created policy. Under the “Policy usage” tab, hit the “Attach” button.

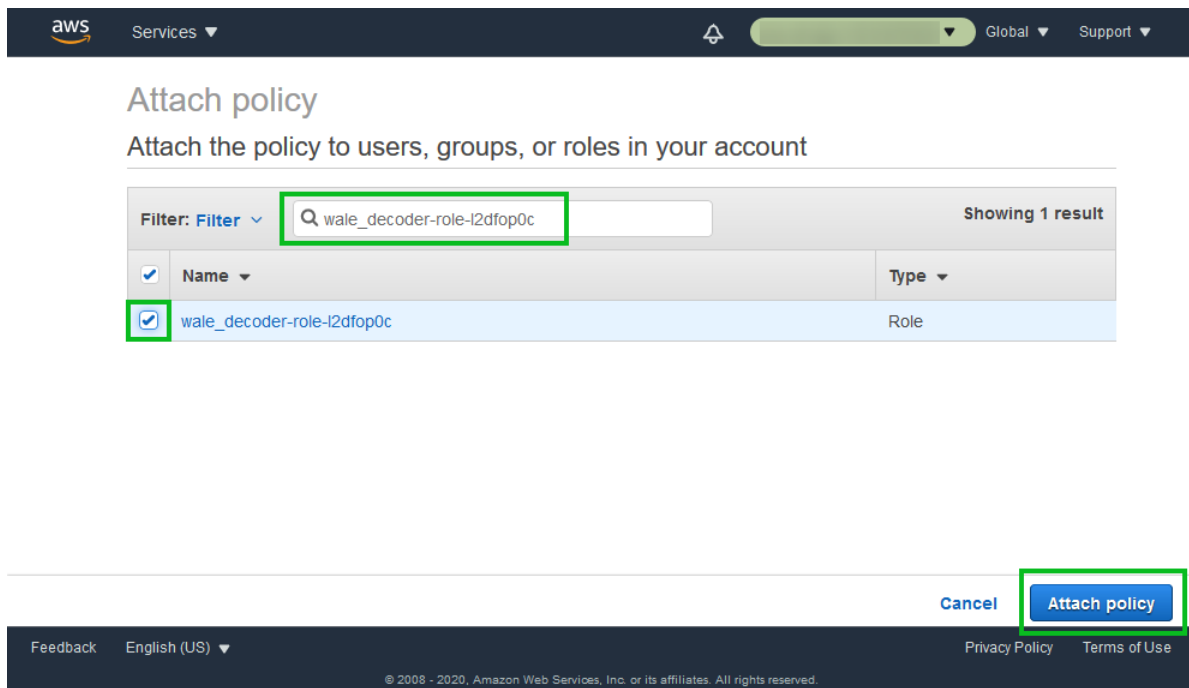


Figure 24: AWS IAM - Attach policy

Search for the lambda function role in the search bar, tick its line and finally hit “Attach policy” button.

If you navigate back to the lambda function role, you will see that 2 policies are attached to the role.

2.5 Follow application execution

Your application is now all setup and ready to run. For each uplink message received on AWS IoT Core, an email will be sent by AWS SNS.

To troubleshoot or watch your application behaviour, several tools come handy.

2.5.1 AWS IoT Core test

Navigate to the AWS IoT Core console. In the navigation pane on the left, click Test. This takes you to the MQTT Test client page, where you can subscribe to an IoT Core topic and watch the data flowing in. You can also manually publish data on any IoT Core topic.

2.5.2 AWS Cloudwatch

Cloudwatch collects the logs of the lambda function. If the function raises an error, Cloudwatch logs will collect and display the function traces.

2.5.3 AWS SNS publish

AWS SNS allows to manually publish a message onto a given topic. This will help investigating if notification emails go to your spam box.

3 Going further

This is it! You have designed and set-up a complete end-to-end application, from the LoRaWAN sensor up to the e-mail notification system.

You can imagine many other usages on top of this infrastructure, such as storing the data sent by the device into AWS S3 or DynamoDB, try the AWS IoT Core Analytics, and even reply back to the device to light an LED.

End of Document