thinxtra
Empowering Internet of Things

sigfox operator

XKIT

amazon web services

INTEGRATION

April 2017

How to integrate your Xkit with AWS IoT Hub

www.thinxtra.com

INTRODUCTION

This is a documentation to help you create callbacks in the Sigfox backend to view parsed messages from the Xkit, on AWS.
To be able to complete the flow, you will need:

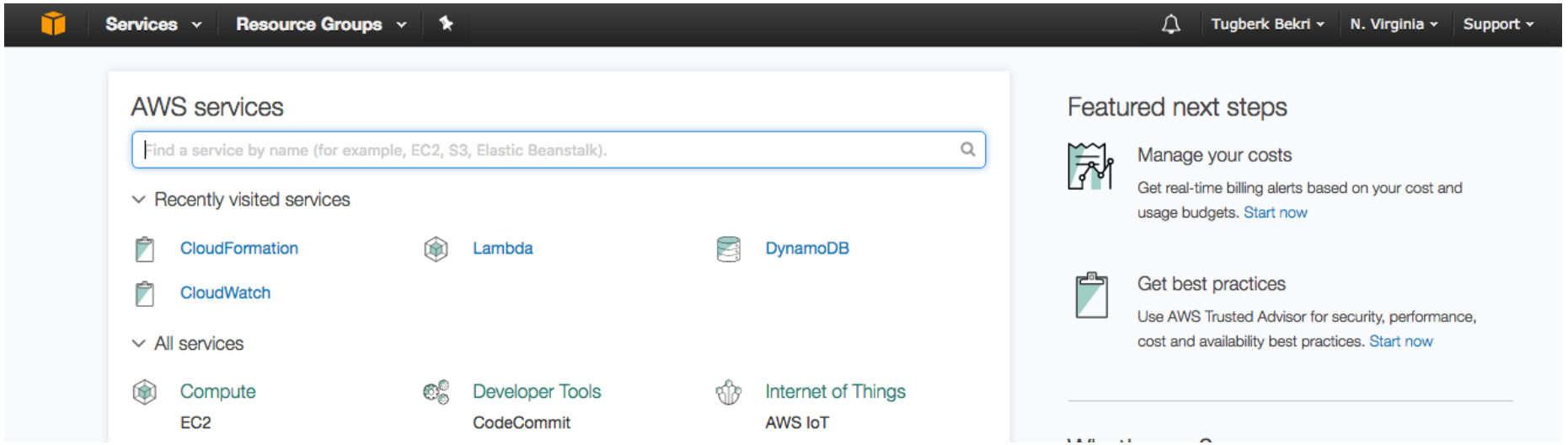
- a registered Thinxtra Xkit on Sigfox backend
- a Sigfox backend account with at least the customer rights
- an AWS account (trial account is enough)

This documentation is separated in the following different sections:

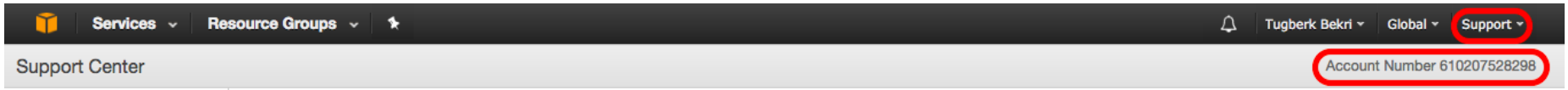
I/ Prepare Amazon Web Services Account
II/ Configure Sigfox callbacks
III/ Check that messages have been received in AWS IoT
IV/ Create Lambda function parse the payload
V/ Push data into DynamoDB

I/ PREPARE AMAZON WEB SERVICES ACCOUNT

STEP 1 Go to https://aws.amazon.com/ and create your free trial account.



STEP 2 Click on “Support” in the top right corner, then “Support Center”.
Copy your account number located in the upper right corner.

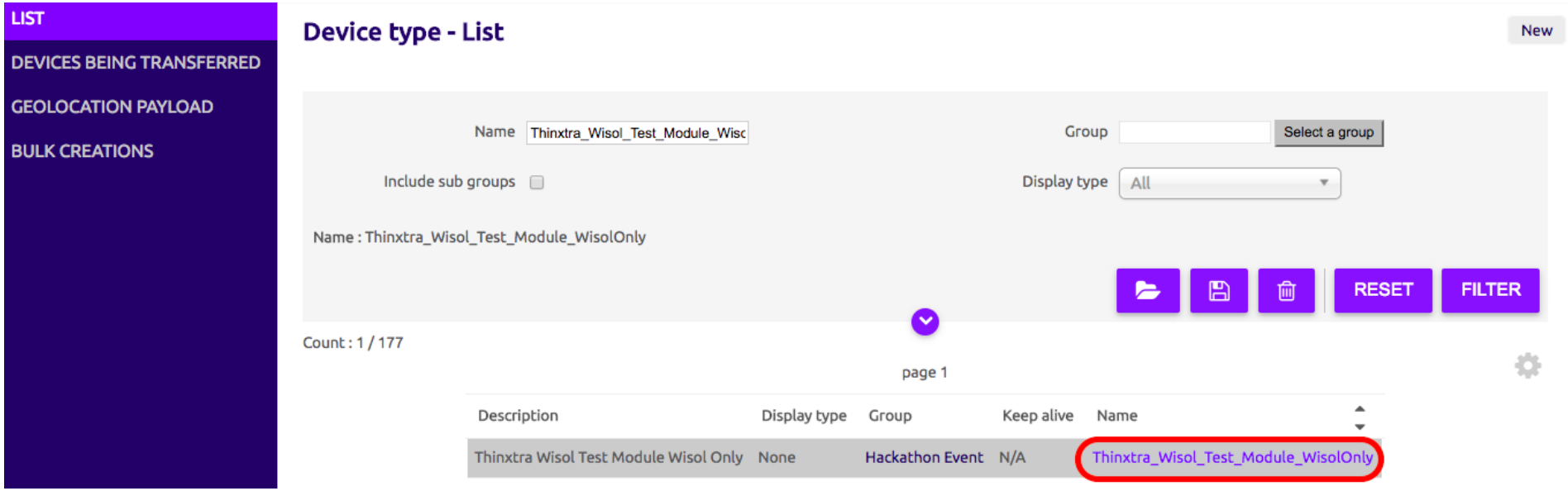


II/ CONFIGURE SIGFOX CALLBACKS

STEP 1 Connect and login to the Sigfox backend at https://backend.sigfox.com

STEP 2 Select “Device Type” tab on the top menu.

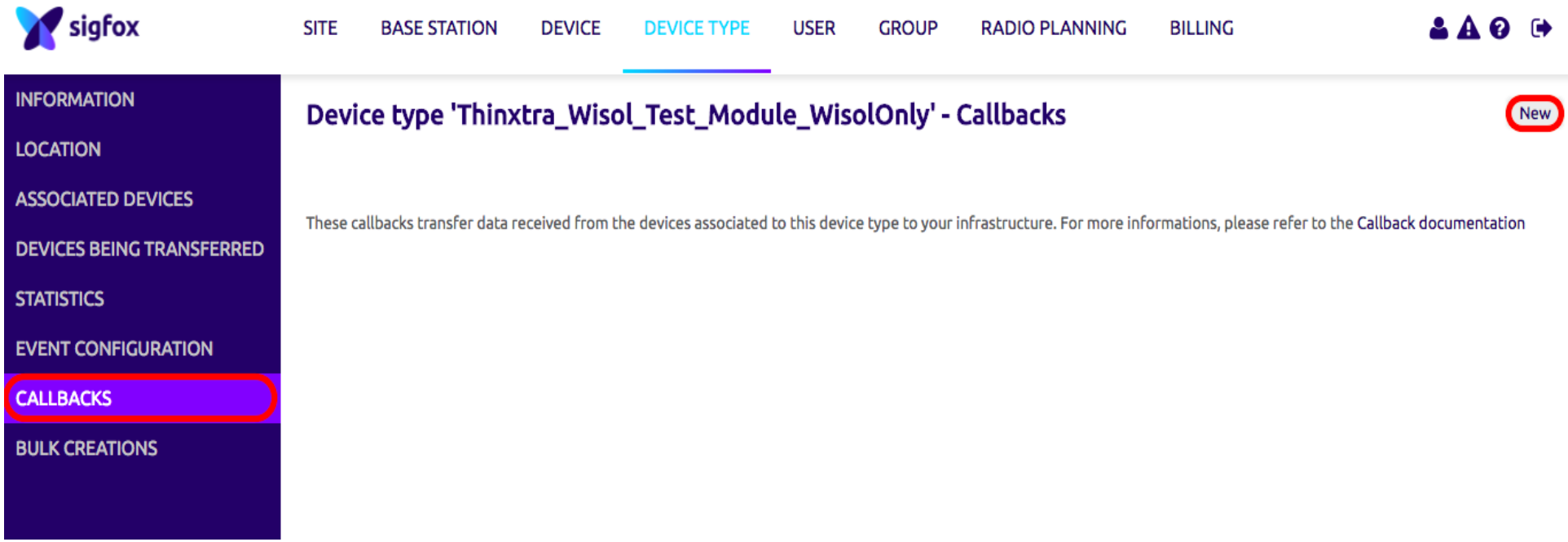
STEP 3 Select or type in your device type from the “Name” column and click on it.



II/ CONFIGURE SIGFOX CALLBACKS

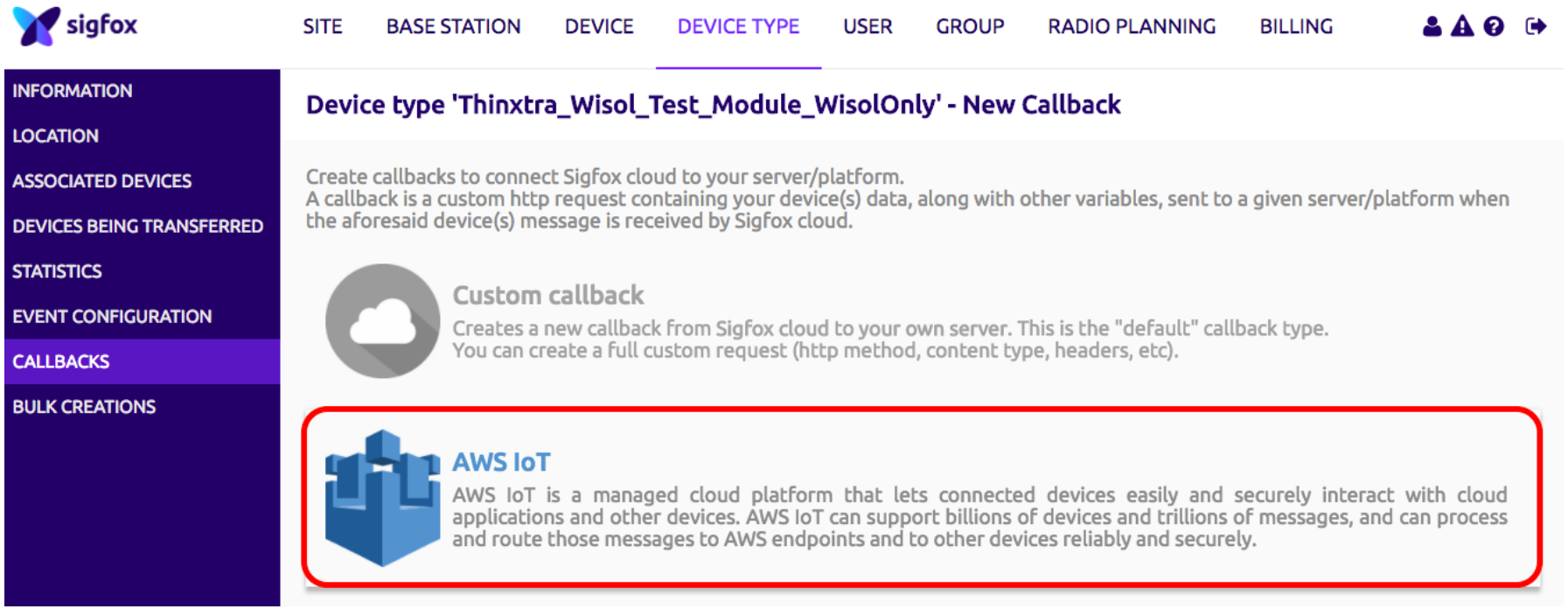
STEP 4 Click on “Callbacks” on the left side menu.

STEP 5 Click on “New” on the top right corner.



II/ CONFIGURE SIGFOX CALLBACKS

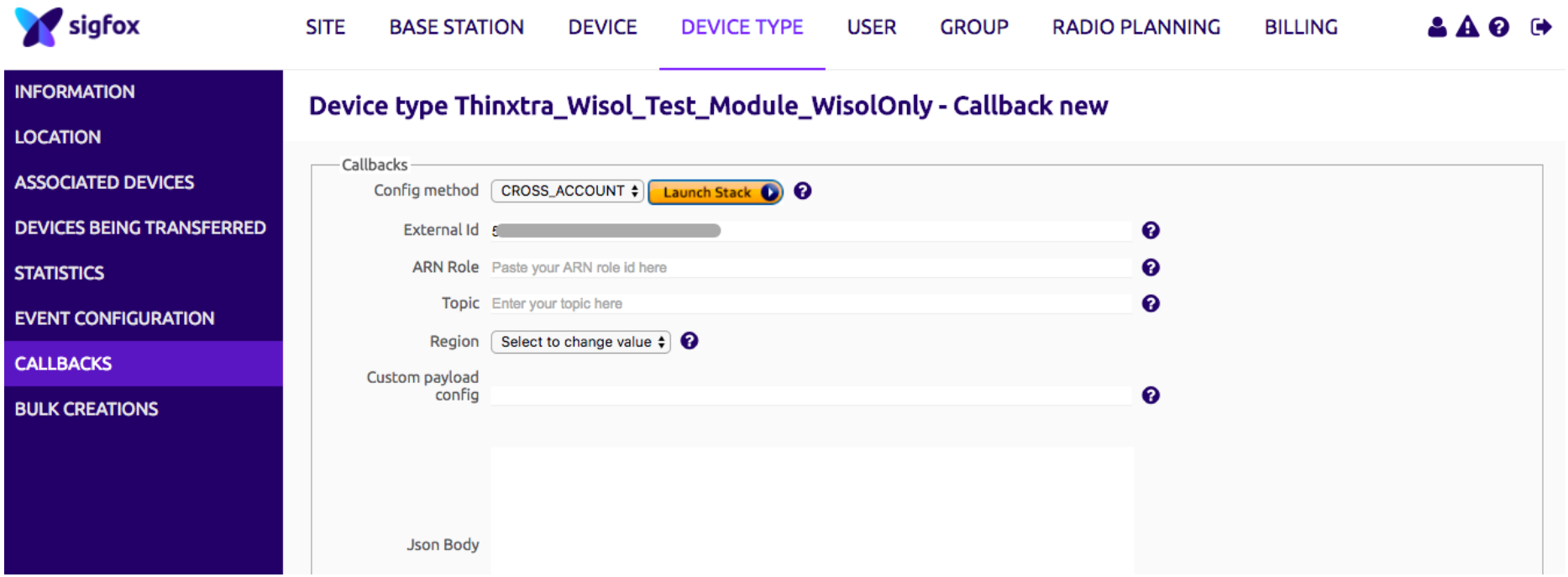
STEP 6 Choose AWS IoT as a new callback.



You can also work on AWS Kinesis if you have an account. For the sake of this tutorial, we will be using AWS IoT.

II/ CONFIGURE SIGFOX CALLBACKS

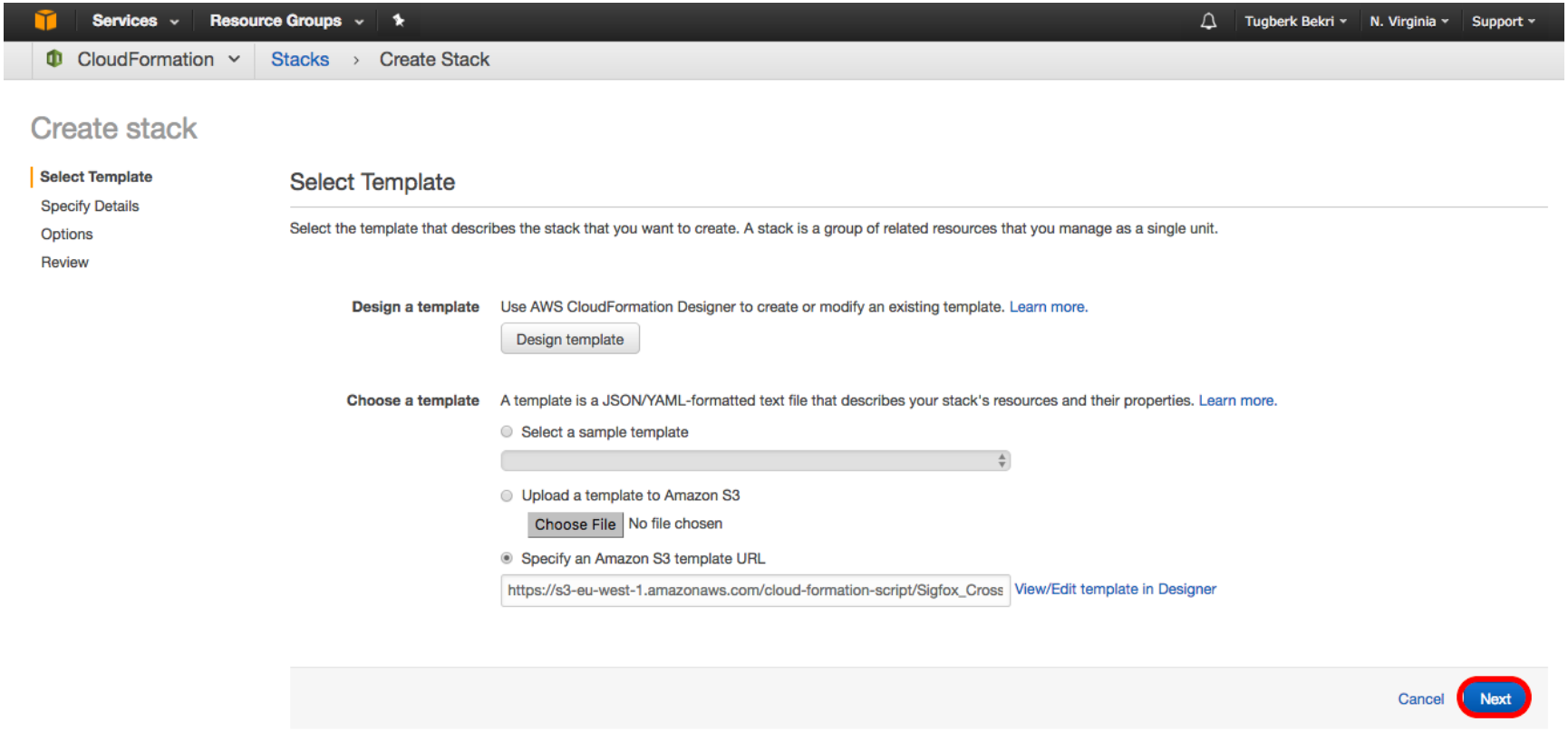
STEP 7 Copy the “External Id” given to you in your clipboard.



STEP 8 Click on “Launch Stack”.

II/ CONFIGURE SIGFOX CALLBACKS

STEP 9 You have been redirected to the AWS CloudFormation console, click on “Next”.

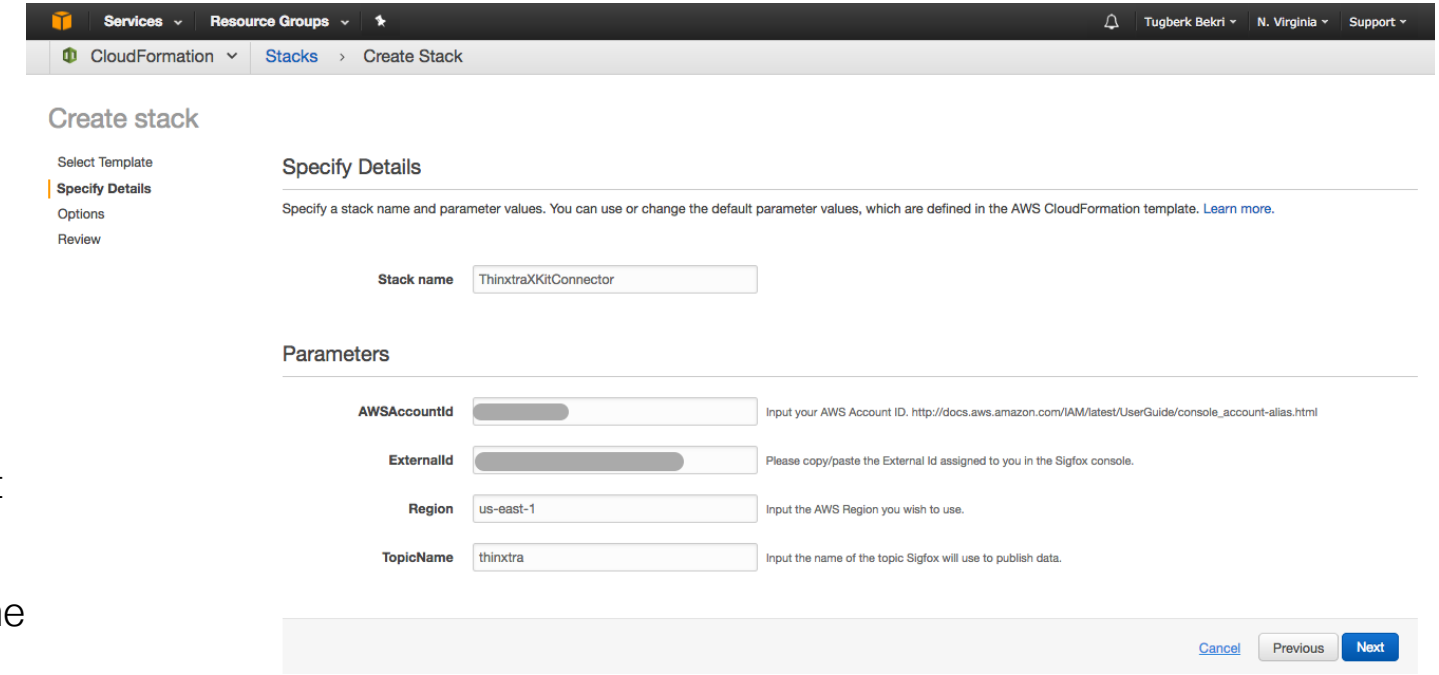


II/ CONFIGURE SIGFOX CALLBACKS

STEP 10

Enter the following inputs:

- *Stack name*: choose a meaningful name
- *AWSAccountId*: paste the Id you copied earlier
- *External Id*: paste the Id off of your Sigfox backend
- *Region*: enter the region name that is located your URL
- *Topic name*: choose the topic name you wish to send data to

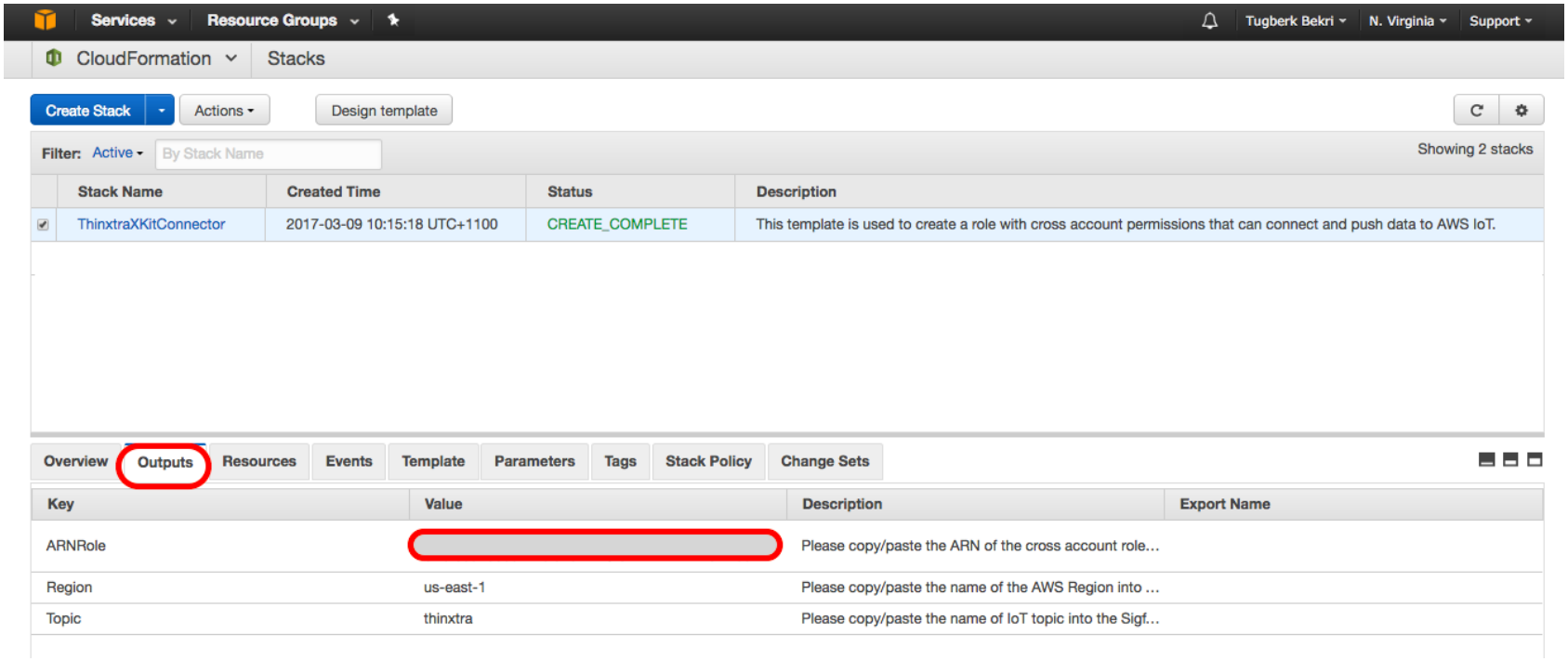


Click on “Next”.

II/ CONFIGURE SIGFOX CALLBACKS

STEP 11 You can customize the next screen optionally, or click “Next”, then the acknowledgment box and “Create”.

STEP 12 Wait for the stack creation completion. Click on “Outputs”. Copy the value of “ARNRole”, “Region” and “Topic”.



II/ CONFIGURE SIGFOX CALLBACKS

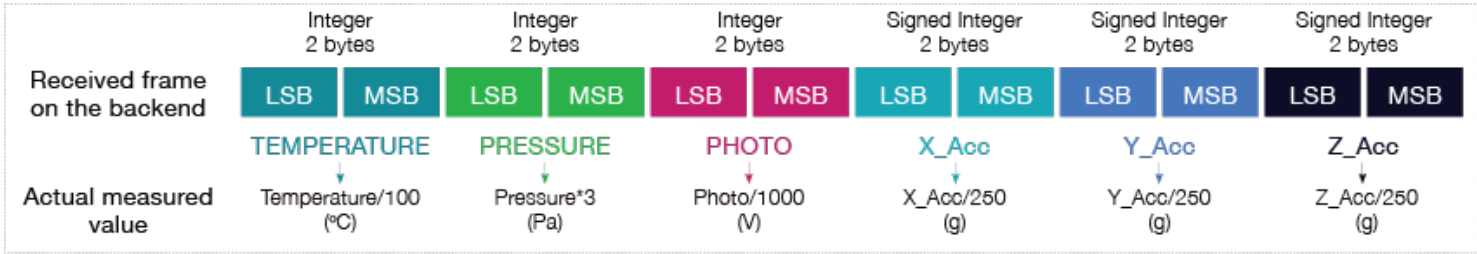
STEP 13

Go back to the Sigfox backend and fill in the following fields:

- ARN Role: paste the value you copied in AWS CloudFormation
- Topic: paste the value you copied earlier
- Region: choose the region from your Cloud Formation stack
- Custom payload config:

```
int1::uint:16:little-endian int2::uint:16:little-endian int3::uint:16:little-endian int4::int:16:little-endian
int5::int:16:little-endian int6::int:16:little-endian
```

Essentially, this payload configuration separates the bytes of the payload into 6 defined data sections as below. Note that the payload is just separated and not decoded yet.



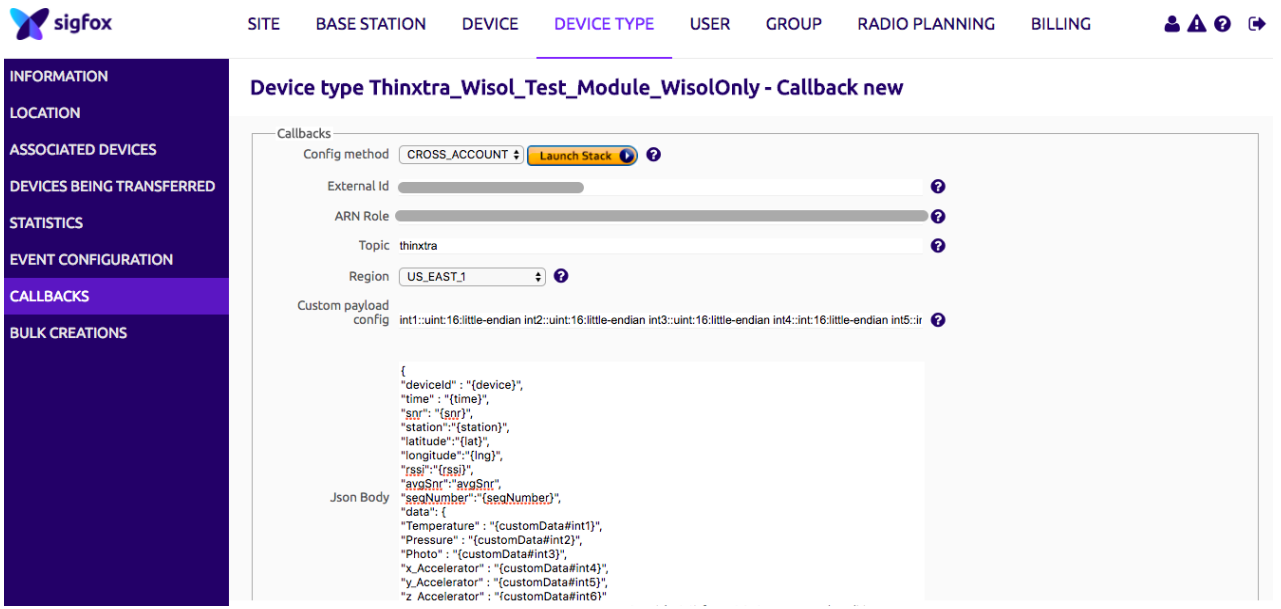
Convert the received values on the backend to the actual measured values

II/ CONFIGURE SIGFOX CALLBACKS

STEP 13 (continued)

- Json Body:

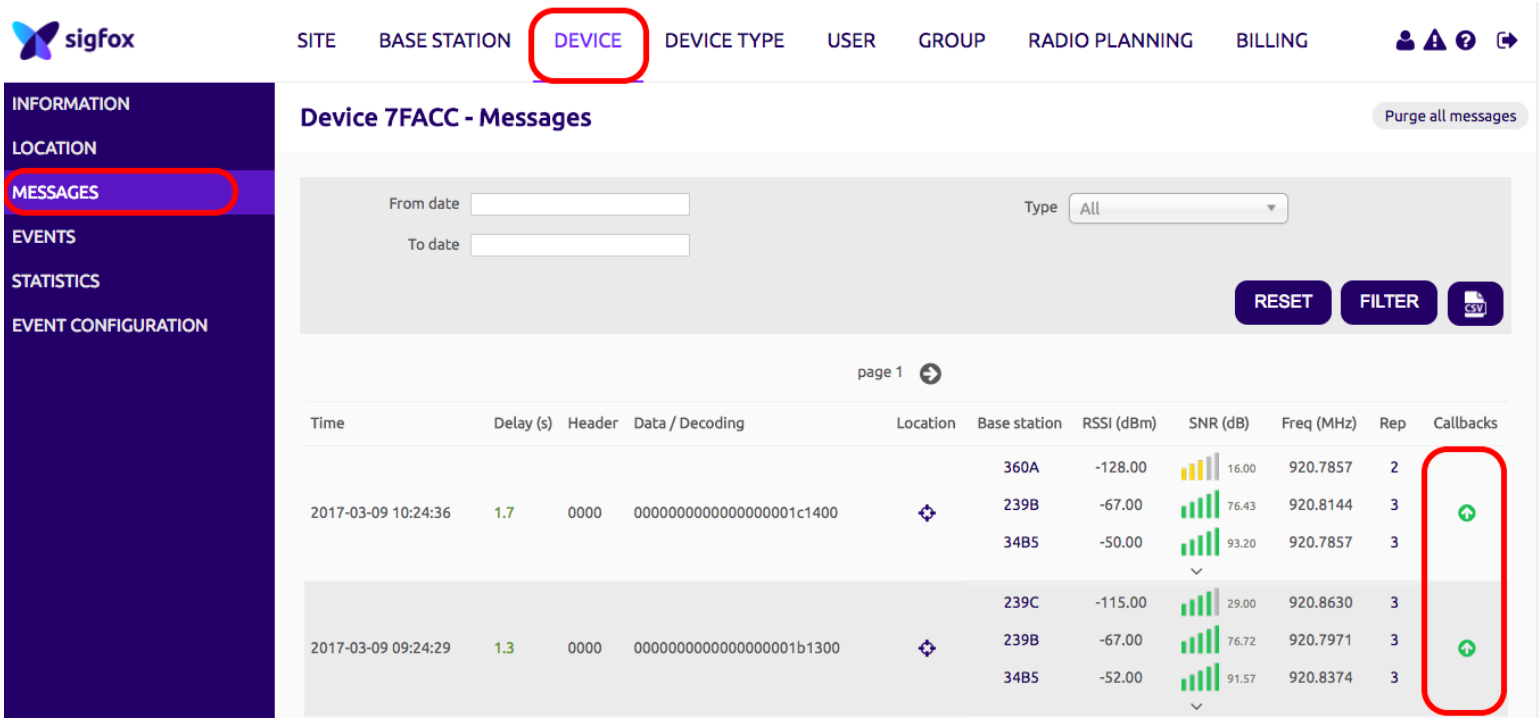
```
{
"deviceId" : "{device}",
"time" : "{time}",
"snr": "{snr}",
"station":"{station}",
"latitude":"{lat}",
"longitude":"{lng}",
"rssi":"{rssi}",
"avgSnr":"{avgSnr}",
"seqNumber":"{seqNumber}",
"data": {
"Temperature" : "{customData#int1}",
"Pressure" : "{customData#int2}",
"Photo" : "{customData#int3}",
"x_Accelerator" : "{customData#int4}",
"y_Accelerator" : "{customData#int5}",
"z_Accelerator" : "{customData#int6}"
}
}
```



III/CHECK THAT MESSAGES HAVE BEEN RECEIVED IN AWS IOT

STEP 1 To check if your callback went through, go into “Device” tab, enter your device name .

STEP 2 Click on “Messages” and check if there is a green arrow on the right side of your callback. It is a confirmation that your callback went through without issues.

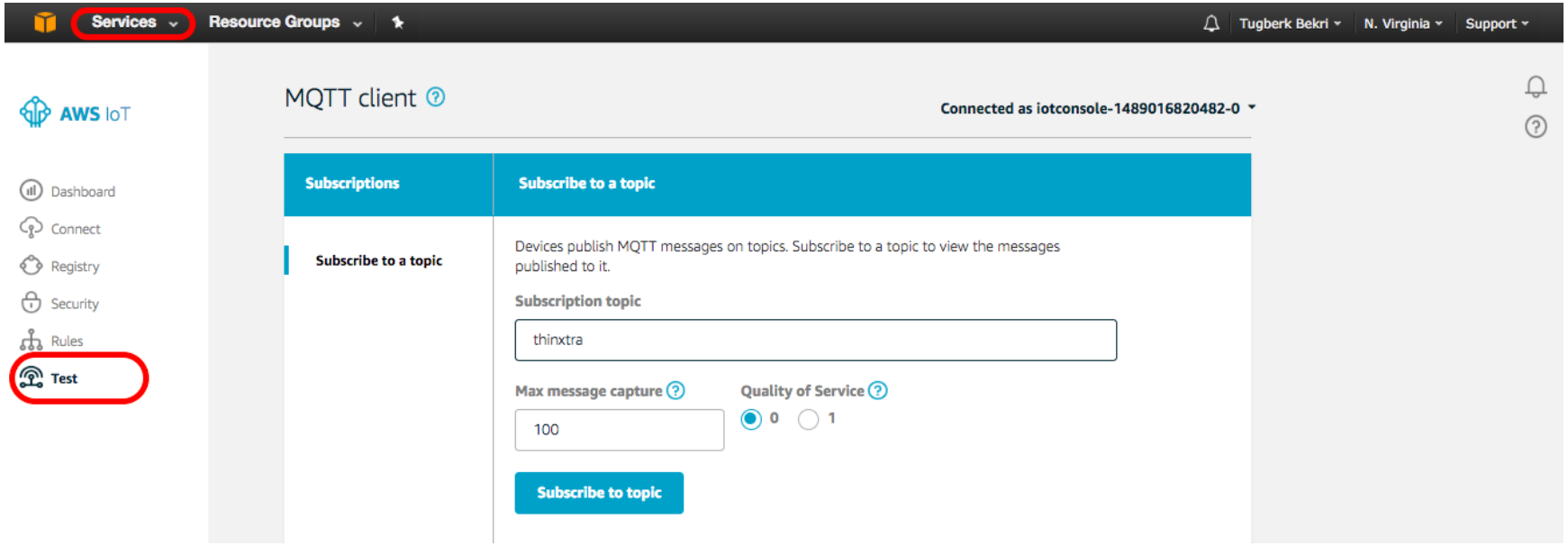


III/CHECK THAT MESSAGES HAVE BEEN RECEIVED IN AWS IOT

STEP 3 Go back into AWS IoT. Go into “Services” in the upper left corner.

STEP 4 Click on AWS IoT, then “Get started”.

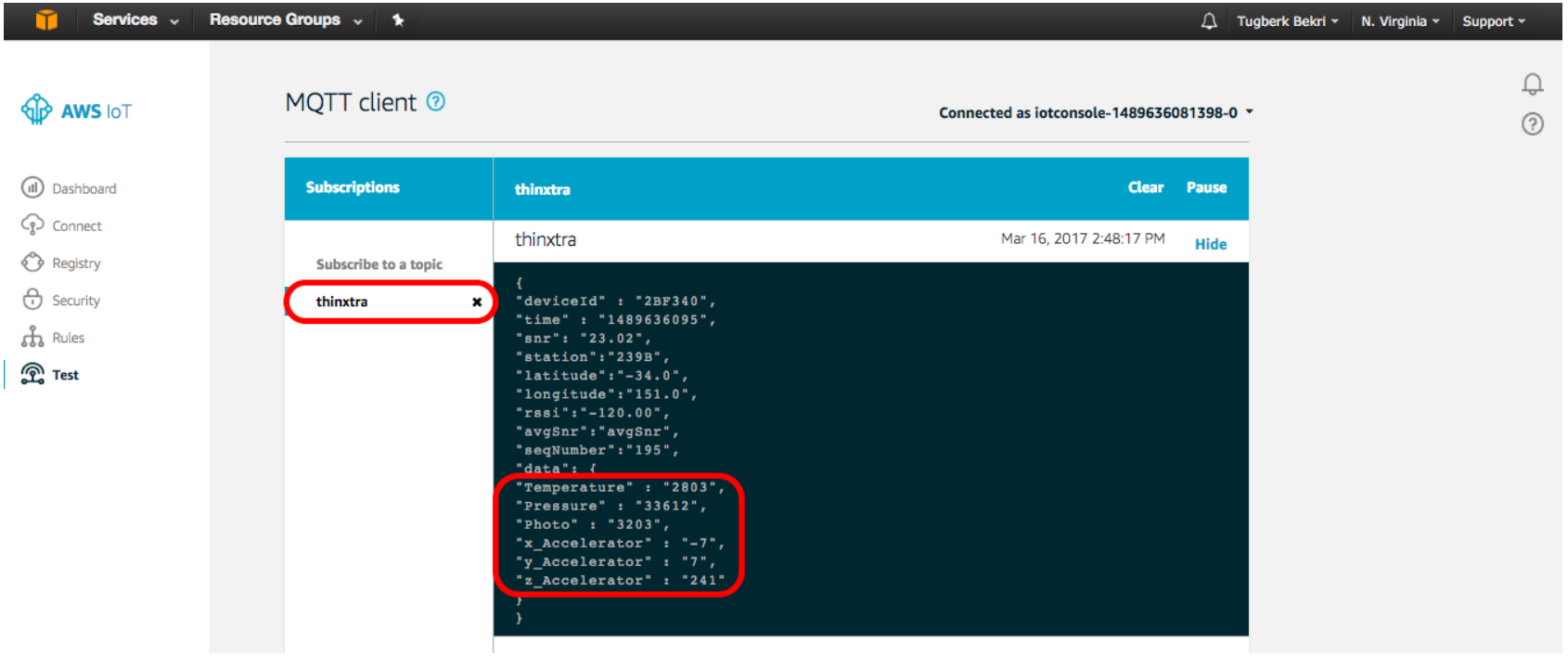
STEP 5 On the left side menu, you will find “Test”, click on it.



III/ CHECK THAT MESSAGES HAVE BEEN RECEIVED IN AWS IOT

STEP 6 Enter the subscription topic you previously chose in your Sigfox callback. Click on “Subscribe to topic”.

STEP 7 Send a message through your Xkit. Click on your subscription topic. You will see your undecoded message.



IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

In this section, you will create a Lambda function that will decode the payload and push it in a Dynamo Database.
By referring back to the “X-kit configuration”, we know that we need to:

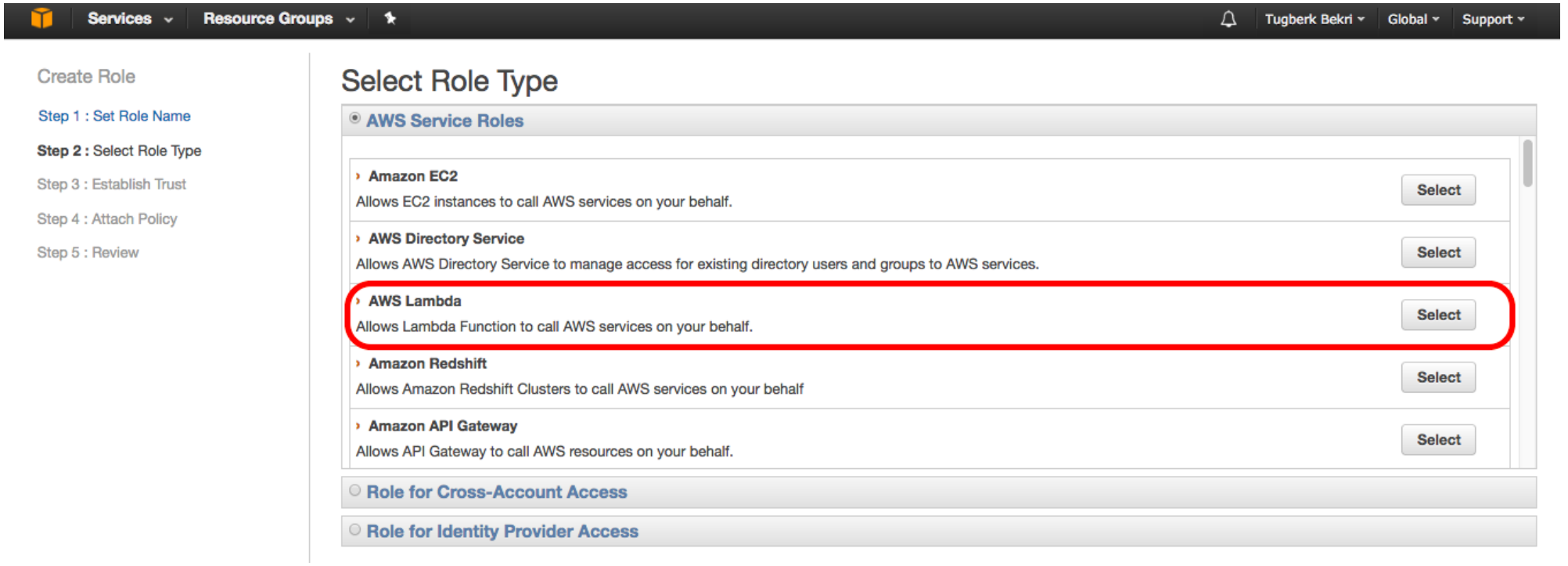
- divide the Temperature data by 100
- multiply the Pressure data by 3
- divide the Photo data by 1000
- divide each of the Accelerator data by 250
- decode the Unix timestamp to a “human-readable” time

STEP 1 Go into “Services” in the top left corner and type in “IAM”.

STEP 2 Click on “Roles” on the left side menu and click on “Create New Role”.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 3 Add a name for your new role. Select “AWS Lambda” as a Role Type.



IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 4 Attach the following policies:

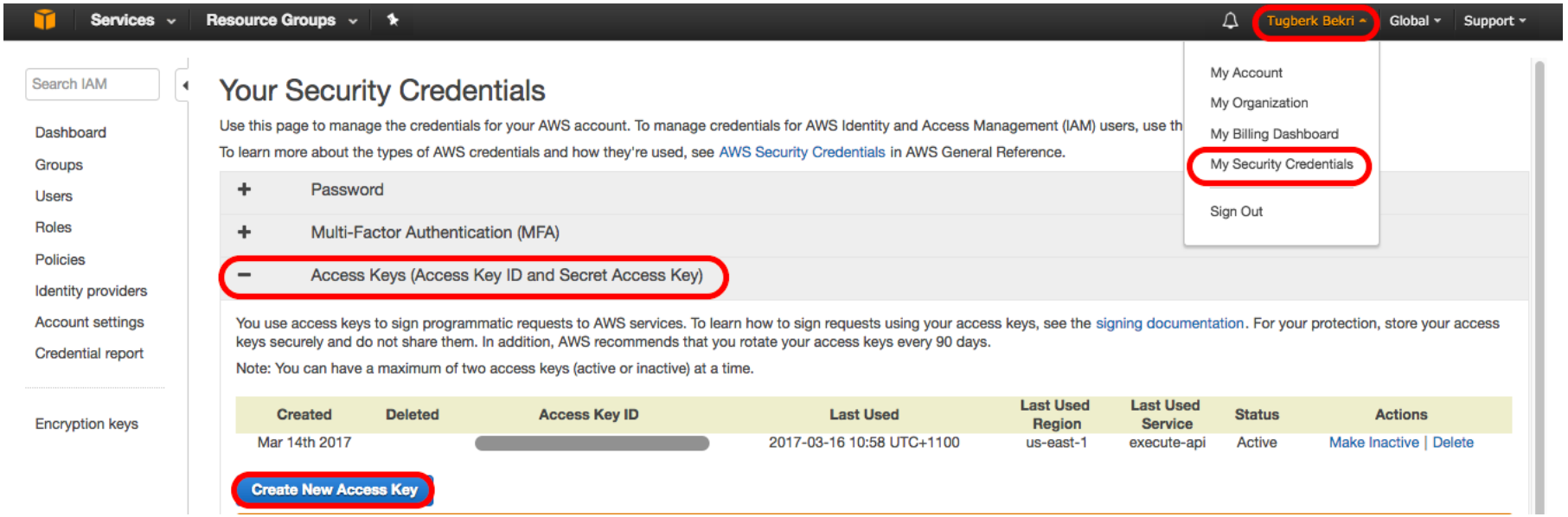
Policy Name
AmazonDynamoDBFullAccess
AWSLambdaDynamoDBExecutionRole
AWSIoTDataAccess
AWSIoTFullAccess

STEP 5 Press “Next” and “Create Role.

You have now created a special IAM role that will allow your function to use the AWS IoT service and write into a Dynamo DB

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 6 Go into your account tab in the upper right corner, click on “My Security Credentials”, then click on “Access Keys” and “Create a new access key”.



STEP 7 Save your access key and secret key in a safe place. You will need them in the next step.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 8 You will now need to compute your IoT endpoint. To do so, you will need to download the Command Line Interface corresponding to your running OS. Follow the option that suits you best.

Option A: if you have pip (a package manager for Python), follow the steps at:
http://docs.aws.amazon.com/cli/latest/userguide/installing.html

Option B: if you don't have pip, following the steps at:

- For Linux, Unix, macOS: http://docs.aws.amazon.com/cli/latest/userguide/awscli-install-bundle.html
- For Windows: http://docs.aws.amazon.com/cli/latest/userguide/awscli-install-windows.html#install-msi-on-windows

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 9 Now that your AWS Command Line Interface is set up, go into your console/terminal and type in:

```
aws configure
```

STEP 10 Your console will ask you for your access key. Add it and press enter.

STEP 11 Your console will then ask you for your Secret Access Key. Add it and press enter.

STEP 12 Your console will then ask you for your Region. Add it and press enter.

STEP 13 Type in the following:

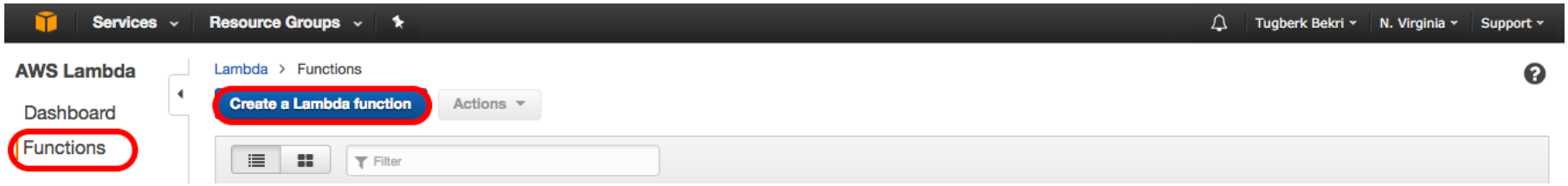
```
aws iot describe-endpoint
```

STEP 14 Copy your AWS IoT endpoint.

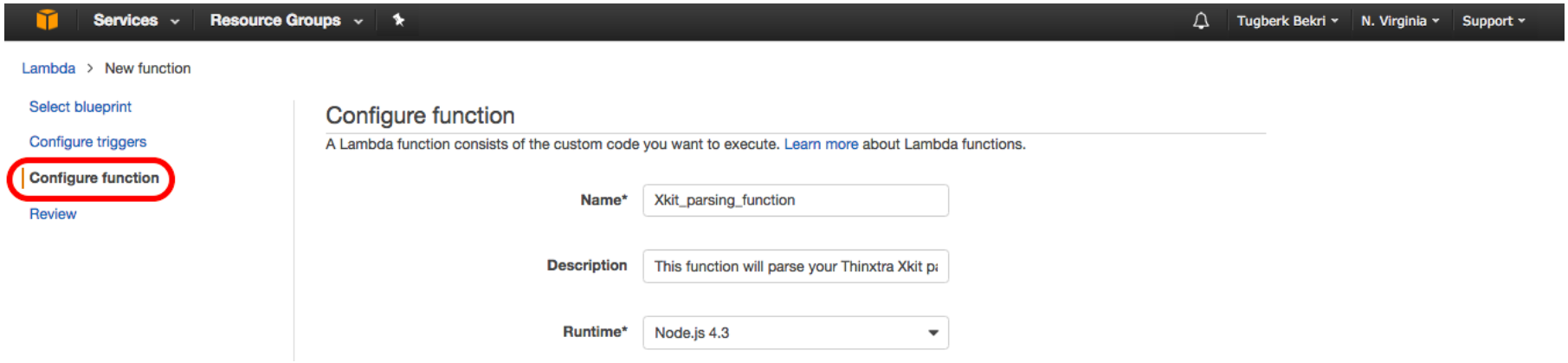
IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 15 Go back into your AWS IoT console. Go into “Services” at the top right and search for “AWS Lambda”.

STEP 16 Go into “Functions” and “Create a Lambda function”.



STEP 17 Click on “Configure function” on the left-side menu. Add a new name for your function and select “Node.js 4.3”.



IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 18 Write your function in the code section as followed.

You can download the full code at: http://www.thinxtra.com/download/6078/

```
console.log('Loading event');
var AWS = require('aws-sdk');
var dynamodb = new AWS.DynamoDB();
var iotData = new AWS.IotData({endpoint: 'your_AWS_endpoint_you_copied_earlier'});

//make sure your replace your_AWS_endpoint_you_copied_earlier with the value you copied just before

exports.handler = function(event, context) {
    console.log("Request received:\n", JSON.stringify(event));
    console.log("Context received:\n", JSON.stringify(context));

    var tableName = event.device;
    var time = event.time;
    var Temperature = event.data.Temperature;
    var Pressure = event.data.Pressure;
    var Photo = event.data.Photo;
    var x_Accelerator = event.data.x_Accelerator;
    var y_Accelerator = event.data.y_Accelerator;
    var z_Accelerator = event.data.z_Accelerator;
    var Accelerator = event.data.Accelerator;
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
function returnTime(value){
        return new Date((value)*1000 + 39600000);
    }

//replace 39600000 with the Epoch time offset according to your timezone. Here it is set to Sydney time
offset.

    timedecode = '' +returnTime(time).toLocaleString();
    Temperature = (Temperature)/100;
    Pressure = (Pressure)*3;
    Photo = (Photo)/1000;
    x_Accelerator = (x_Accelerator)/250;
    y_Accelerator = (y_Accelerator)/250;
    z_Accelerator = (z_Accelerator)/250;
    Accelerator = Math.sqrt(x_Accelerator * x_Accelerator + y_Accelerator * y_Accelerator + z_Accelerator *
z_Accelerator);

var payloadObj={ "state": {
                "reported": {
                "device": event.device,
                "time": event.time,
                "timedecode": timedecode,
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
                "snr": event.snr,
                "avgSnr": event.avgSnr,
                "station": event.station,
                "Temperature": Temperature,
                "Pressure": Pressure,
                "Photo": Photo,
                "x_Accelerator": x_Accelerator,
                "y_Accelerator": y_Accelerator,
                "z_Accelerator": z_Accelerator,
                "Accelerator": Accelerator
                },
                }
};

    var paramsUpdate = {
        thingName : event.deviceId,
        payload : JSON.stringify(payloadObj)
    };

 //This function will update the Device Shadow State

    iotData.updateThingShadow(paramsUpdate, function(err, data) {
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
 if (err){
        console.log("Error in updating the Thing Shadow");
        console.log(err, err.stack);
            }
    });

//This function will store the message in a dynamoDB table
    dynamodb.putItem({
            "TableName": event.deviceId,
            "Item": {
                "deviceId": {
                    "S": event.deviceId
                },
                "time": {
                    "S": event.time
                },
                "timedecode": {
                    "S": timedecode.toString()
                },
                "Temperature": {
                    "S": Temperature.toString()
                },
                "Pressure": {
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

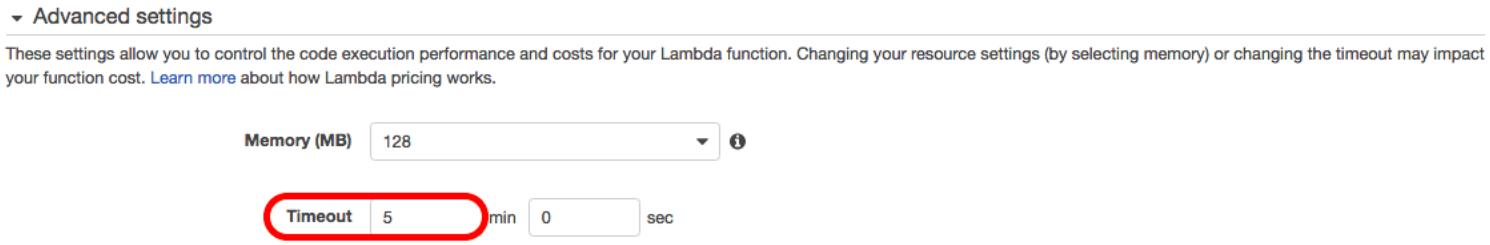
```
            "S": Pressure.toString()
        },
        "Photo": {
            "S": Photo.toString()
        },
        "x_Accelerator": {
            "S": x_Accelerator.toString()
        },
        "y_Accelerator": {
            "S": y_Accelerator.toString()
        },
        "z_Accelerator": {
            "S": z_Accelerator.toString()
        },
        "Accelerator": {
            "S": Accelerator.toString()
        }

    }
},
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
        function(err, data){
            if (err) {
                context.fail('ERROR: Dynamo failed: ' + err);
            } else {
        console.log('Dynamo Success: ' + JSON.stringify(data, null, ' '));
                context.succeed('SUCCESS');
            }
        });
};
```

STEP 19 Set role as “Choose an existing role”, choose the IAM role name you’ve created previously.

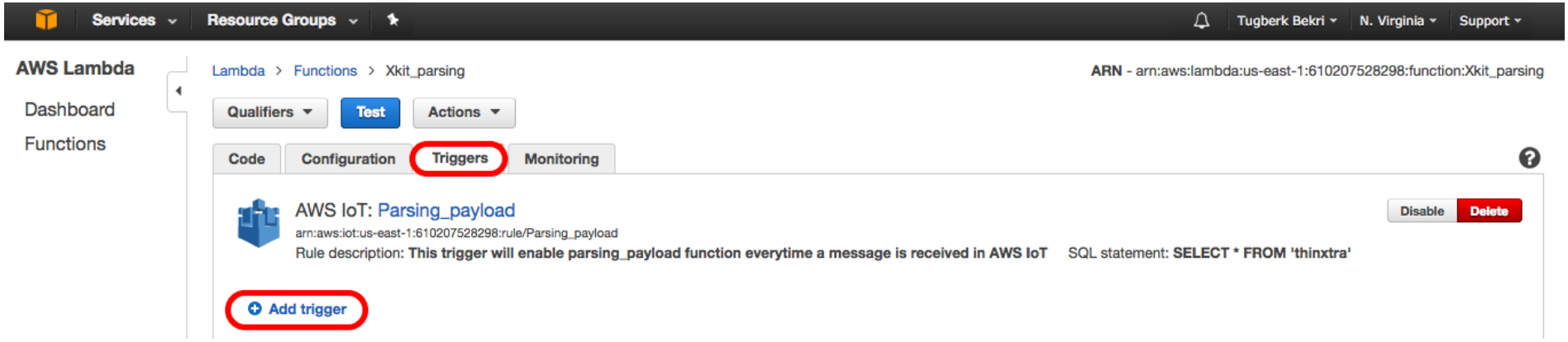
STEP 20 In the advanced settings, set your timeout to 5min.



IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 21 Click on “Create”.

STEP 22 Now click on your function. Click on “Triggers” tab, then “Add a trigger”.



STEP 23 Choose “AWS IoT” as an input.

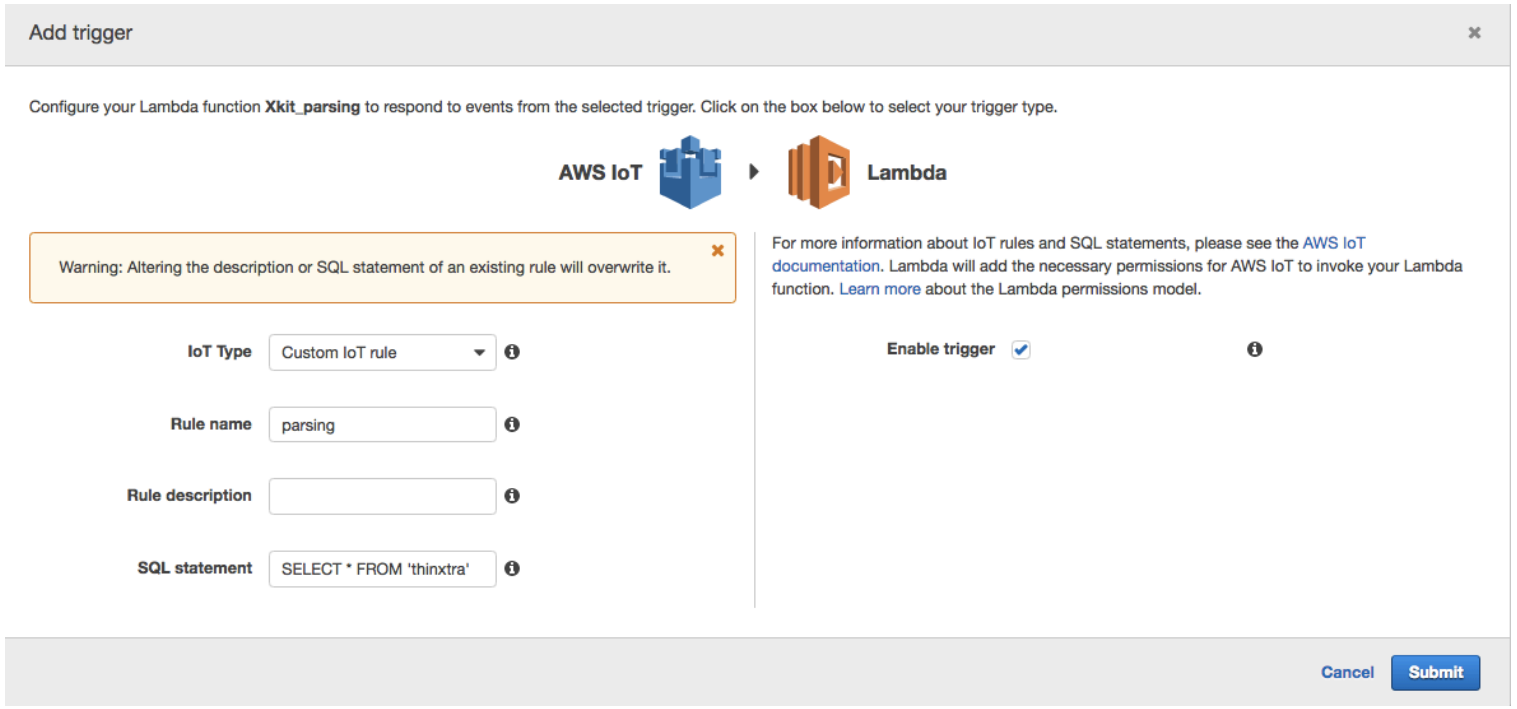
STEP 24 Choose “Custom IoT rule” as IoT Type.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 25 Add a rule name. Enter the following SQL statement and click on “Submit”.

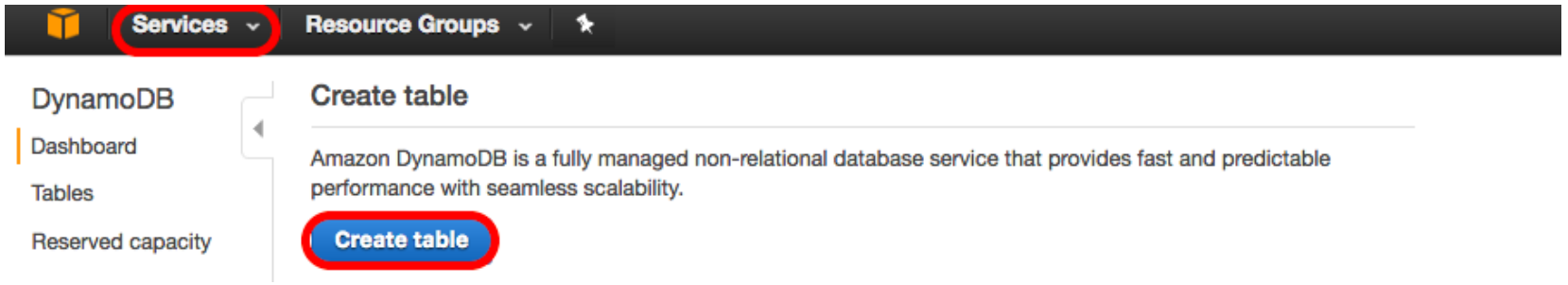
```
SELECT * FROM 'your_topic'
```

Make sure to change your_topic with the topic you entered in your callback in the Sigfox backend.



V/ PUSH DATA INTO DYNAMODB

STEP 1 Go to services and click on “DynamoDB”. Click on “Create table”.



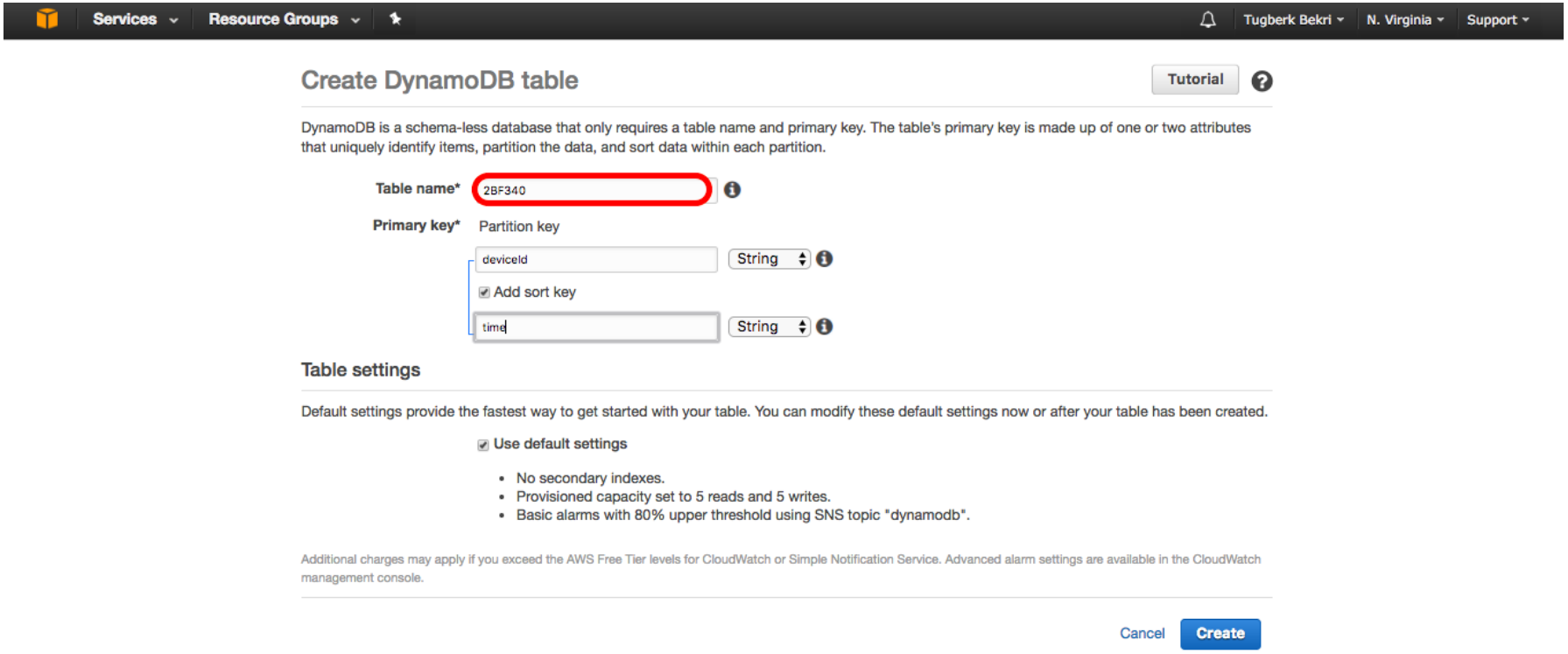
STEP 2 Add your deviceId (Xkit Id on the Sigfox backend) as a Table name.

It is very important to name your table with your deviceId since it is how we configured our Lambda function.

V/ PUSH DATA INTO DYNAMODB

STEP 3 Add ‘deviceId’ as a primary key and ‘time’ as a sort key.

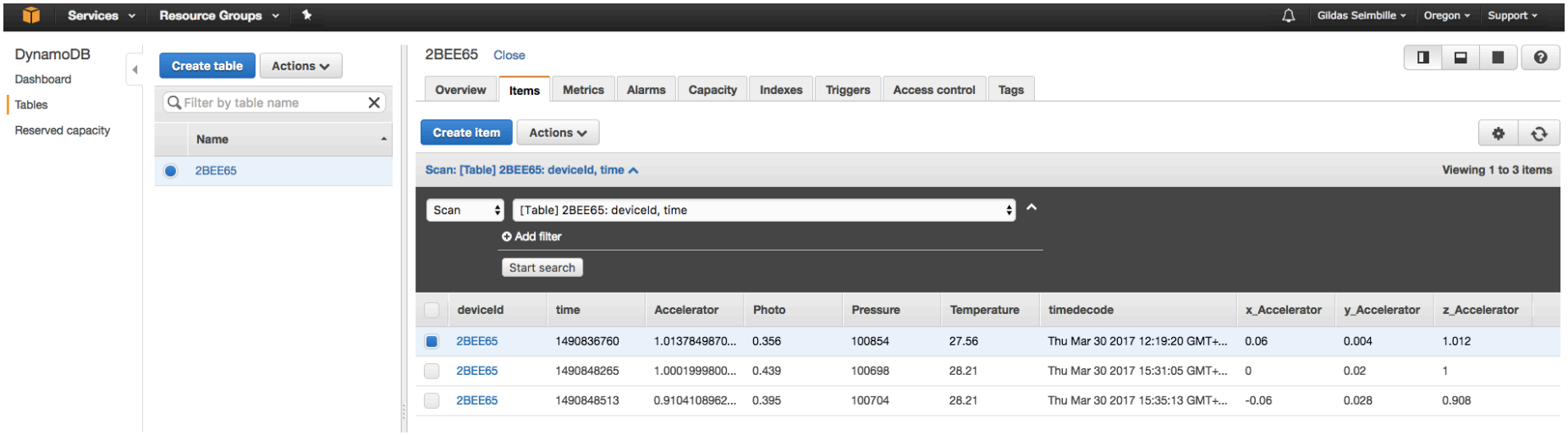
Click on “Create”.



V/ PUSH DATA INTO DYNAMODB

STEP 4 Once you have created your table, click on it and go into the items Tab.

STEP 5 Send a message from your Xkit, you should be seeing the parsed payload coming in.



V/ PUSH DATA INTO DYNAMODB

STEP 6 Click on your deviceId. You should be able to see your latest full decoded message.

```
Accelerator String : 0.9529491067208153
deviceid String : 2BF340
Photo String : 1.074
Pressure String : 100764
Temperature String : 29.2
timedecode String : Thu Mar 16 2017 15:44:43 GMT+0000 (UTC)
timestamp String : 1489639483
x_Accelerator String : -0.032
y_Accelerator String : 0.028
z_Accelerator String : 0.952
```

You have now sent your message through the Sigfox backend to AWS IoT Hub. You parsed your payload via a Lambda function and pushed the data to a DynamoDB table.

If you wish to move further with the integration and visualize the message in a more graphic matter, you can push your DynamoDB data to AWS ElasticSearch service coupled with a third-party Visualisation dashboard tool.


thinxtra
Empowering Internet of Things
sigfox
operator
www.thinxtra.com/xkit/
More infos on AWS website:
aws.amazon.com/fr/blogs/iot/connect-your-devices-to-aws-iot-using-the-sigfox-network/