Prototype Study Report

**Real-time Analytical Monitoring Application (RAM)**

|  |  |
| --- | --- |
|  |  |
| Document Reference: | 1.0 |
| Project: | Real-time Analytical Monitoring Application (RAM) |
| Document Title: | Prototype Study Report |
| Version: | Draft |
| Date: | 06 May 2017 |
| Author: | GAO ZHIYU |
| Reviewers: |  |
| **©** 2017  The information contained in this document is the property of ISS. The contents must not be copied in whole or in part for purposes other than which it has been supplied without the consent of ISS, or, if it has been furnished under contract to another party, as expressly authorised under that contract, then ISS shall not be liable for any errors or omissions. |  |
| ***Institute of Systems Science, 25 Heng Mui Keng Terrace,***  ***Singapore 119615*** | |

**Document Reference:** FMCA/TPS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Authorised by** | **Date** |
| Draft | 06/05/18 | GAO ZHIYU |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**FOODY Mobile Client Application (RAM)**

**Prototype Survey Report**

**Distribution:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Department** | **Organisation** |
| GAO ZHIYU |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Prototype Study Report Summary 5](#_Toc513393242)

[1 INTRODUCTION. 7](#_Toc513393243)

[2 Environment Setup 8](#_Toc513393244)

[2.1 Java8 8](#_Toc513393245)

[2.2 AWS 8](#_Toc513393246)

[2.3 Windows PC 8](#_Toc513393247)

[3 AWS Component 8](#_Toc513393248)

[3.1 steps and procedures for the Lambda and dynamoDB 8](#_Toc513393249)

[3.1.1 Download the sourcecode 8](#_Toc513393250)

[3.1.2 Building the Deployment File 8](#_Toc513393251)

[3.1.3 Creating the Dynamo Table 8](#_Toc513393252)

[3.1.4 Creating the Lambda Function 9](#_Toc513393253)

[3.1.5 Testing the Lambda Function 9](#_Toc513393254)

[3.2 Set up the Windows PC as the device to IOT core 10](#_Toc513393255)

[3.2.1 Go to IOT core, and click the onboarding 10](#_Toc513393256)

[3.2.2 Click the Get Started of Configuring a Device and it will route the screen below 11](#_Toc513393257)

[3.2.3 Click the Get start and it will route the page below 12](#_Toc513393258)

[3.2.4 choosing the correct OS/toolset and click the next 12](#_Toc513393259)

[3.2.5 Giving a name and click "next step" 12](#_Toc513393260)

[3.2.6 Click the windows to download the toolset, and next steps will become active 13](#_Toc513393261)

[3.2.7 After clicking next steps, it will route to the page with details text below 13](#_Toc513393262)

[3.2.8 To configure and test the device, perform the following steps. 13](#_Toc513393263)

[3.3 EndToEnd UI 14](#_Toc513393264)

[3.3.1 Device -- IOT-- Lambda--DynamoDB 14](#_Toc513393265)

[3.3.1.1 windows acting as the device, sending data to iot core 14](#_Toc513393266)

[3.3.1.2 subscribe the MQTT topic, we are able to find the msgs 15](#_Toc513393267)

[3.3.1.3 IOT set a rule, when the required msg arrived, it will trigger the lambda 15](#_Toc513393268)

[3.3.1.4 The lambda will save the db to dynamoDB 16](#_Toc513393269)

[3.3.2 The lambda will send data back to IOT core 16](#_Toc513393270)

[3.3.2.1 Prepare the test case 16](#_Toc513393271)

[3.3.2.2 Execute the Lambda 16](#_Toc513393272)

[3.3.2.3 Verify IOT core successfully capture the msg 17](#_Toc513393273)

# Prototype Study Report Summary

The below is the high-level summary of problem areas that were included to be addressed as part of prototyping and technology evaluation criteria

Table 1: Summary of the Prototype components

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Requirements**  **Category** | **Problem Statement** | **Technology Solution Proposed and Prototyped** | **Conclusion/Outcomes** |
| 1 | Functional | We need to store the structure data. As RDBMS is not a proper solution to us, so we decide to adopt the NoSQL approach | We adopt the AWS dynamoDB | DynamoDB provide functionality of NoSQL  Easier for us to do the clustering. No need to worry about the maintenance |
| 2 | Functional | We need to retrieve the information from device. | We adopt the IOT core. As we can publish and subscribe the message from topic. | MQTT Protocol Specification is a fit to purpose and easy to use solution for this problem. |
| 3 | Functional | When the msg is derived from the mqtt, we need to perform business actions | We will create the iot core rule for the msg received from the topic and perform the lambda function | Based on the research, IOT core can be used as the event source of lambda, we can create rules for how to route the messages |
| 4 | Functional | We need to process the msg received from mqtt, after the processing, we need to store in the database | We will adopt the lambda to perform the relevant the DB call | Based on the research, dynamoDB can be used the downstream of the DB call. |
| 5 | 5 | After we processed the data, we need to send back to mqtt for real time display | We need to adapt the security (private key / cert) to publish the message back to mqtt. |  |

# INTRODUCTION.

This prototype is to study the feasibility of using AWS related solution to build system for seris Application. A prototype using the AWS core/lambda/ec2/dynamoDB/ solution to be created to evaluate the complexity of the technology.

# Environment Setup

Following tools need to be installed in order to start the react native app development

## Java8

For the lambda, we adopt the Java 8 for the prototype.

## AWS

We need to have aws account. For the prototype, we are using the AWS free tier.

https://aws.amazon.com/free/

## Windows PC

We use the windows PC to act as the device to communicate to the IOT core

# AWS Component

# steps and procedures for the Lambda and dynamoDB

## Download the sourcecode

*https://github.com/gaozhiyu/SEProjectPOC.git*

## Building the Deployment File

To build the lambda application, we need to execute the following Maven command:

mvn clean package shade:shade

Lambda application will be compiled and packaged into a jar file under the target folder.

## Creating the Dynamo Table

Follow these steps to create the Dynamo table:

* Login to [AWS Account](https://aws.amazon.com/)
* Click “DynamoDB” that can be located under “All Services”
* This page will show already created DynamoDB tables (if any)
* Click “Create Table” button
* Provide “Table name” and “Primary Key” with its datatype as “Number”
* Click on “Create” button
* Table will be created

## Creating the Lambda Function

Follow these steps to create the Lambda function:

* Login to [AWS Account](https://aws.amazon.com/)
* Click “Lambda” that can be located under “All Services”
* This page will show already created Lambda Function (if any) or no lambda functions are created click on “Get Started Now”
* “Select blueprint” -> Select “Blank Function”
* “Configure triggers” -> Click “Next” button
* “Configure function”
  + “Name”: SavePerson
  + “Description”: Save Person to DDB
  + “Runtime”: Select “Java 8”
  + “Upload”: Click “Upload” button and select the jar file of lambda application
* “Handler”: com.baeldung.lambda.dynamodb.SavePersonHandler
* “Role”: Select “Create a custom role”
* A new window will pop and will allow configuring IAM role for lambda execution and we need to add the DynamoDB grants in it. Once done, click “Allow” button
* Click “Next” button
* “Review”: Review the configuration
* Click “Create function” button

## Testing the Lambda Function

Next step is to test the lambda function:

* Click the “Test” button
* The “Input test event” window will be shown. Here, we’ll provide the JSON input for our request:

|  |  |
| --- | --- |
|  |  |
|  | {    "id": 1,    "firstName": "John",    "lastName": "Doe",    "age": 30,    "address": "United States"  } |
|  |  |
|  |  |

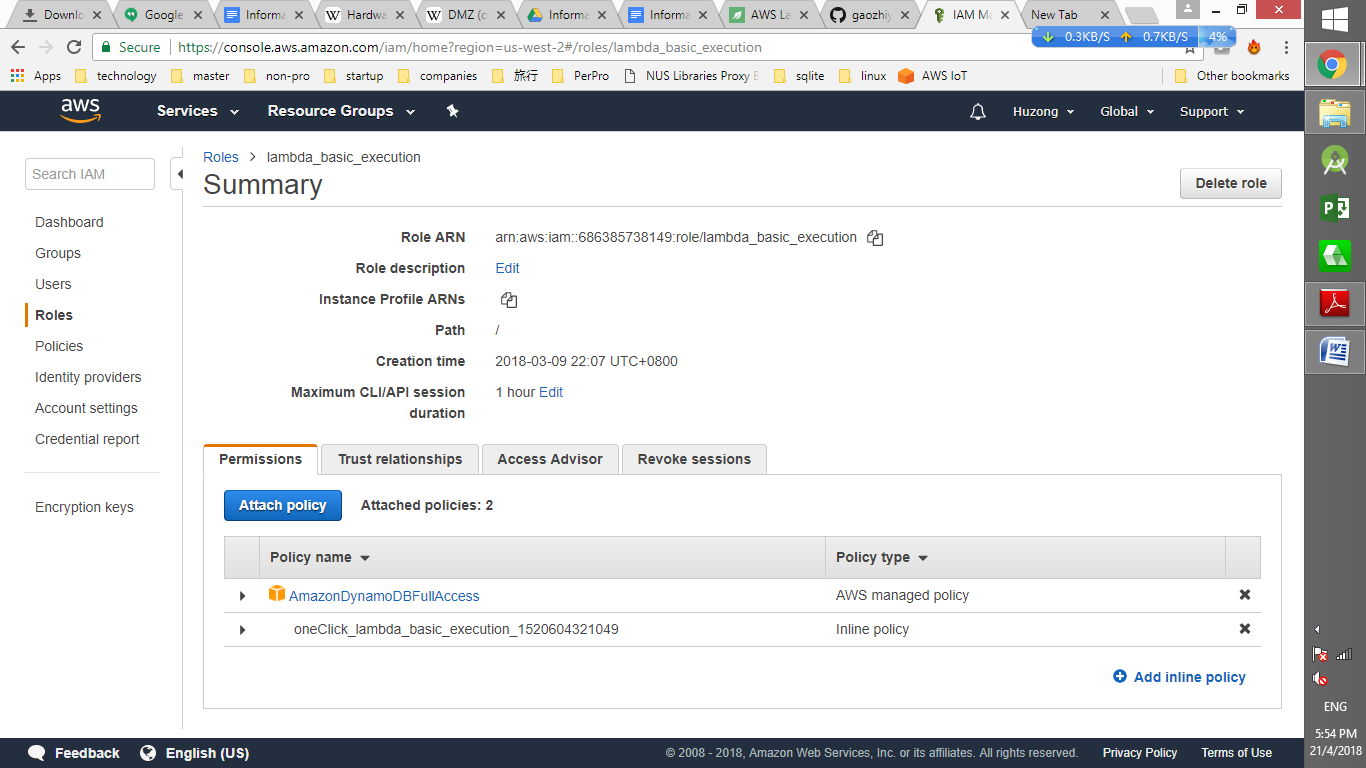
* Click “Save and test” or “Save” button
* The output can be seen on “Execution result” section:

|  |  |
| --- | --- |
|  | {    "message": "Saved Successfully!!!"  } |

* We also need to check in DynamoDB that the record is persisted:
  + Go to “DynamoDB” Management Console
  + Select the table “Person”
  + Select the “Items” tab
  + Here you can see the person’s details which were being passed in request to lambda application
* So the request is successfully processed by our lambda application

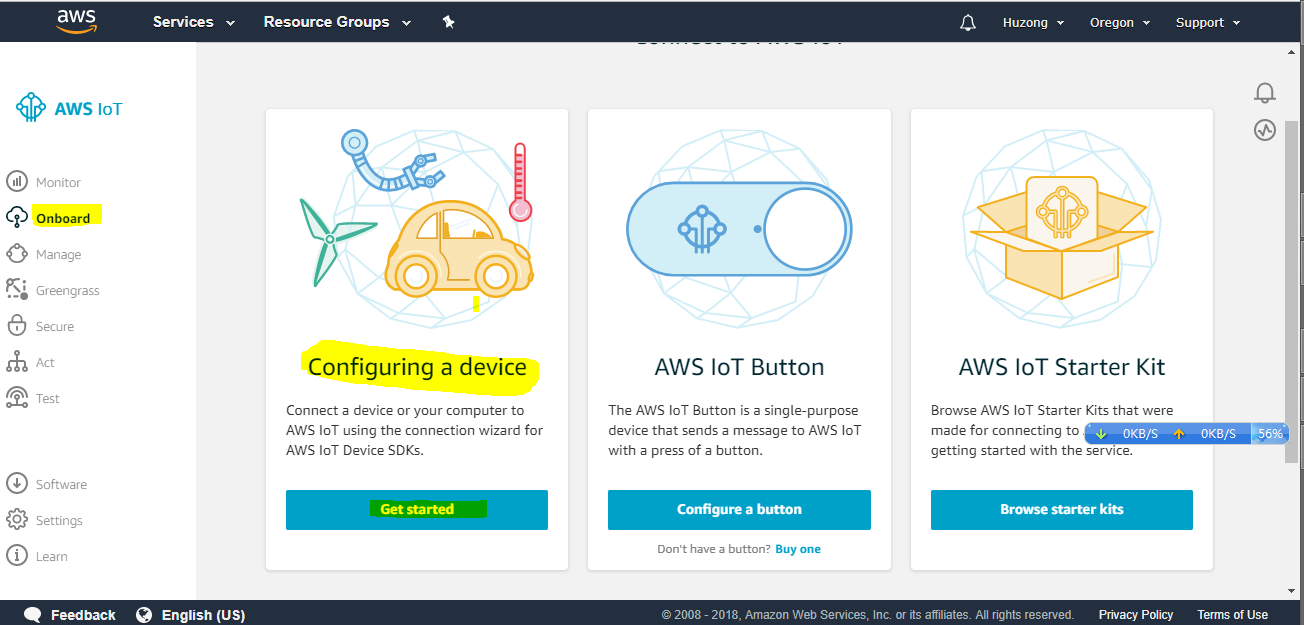
p.s:

The most tricky part is to allow the lambda to access the dynamodb function

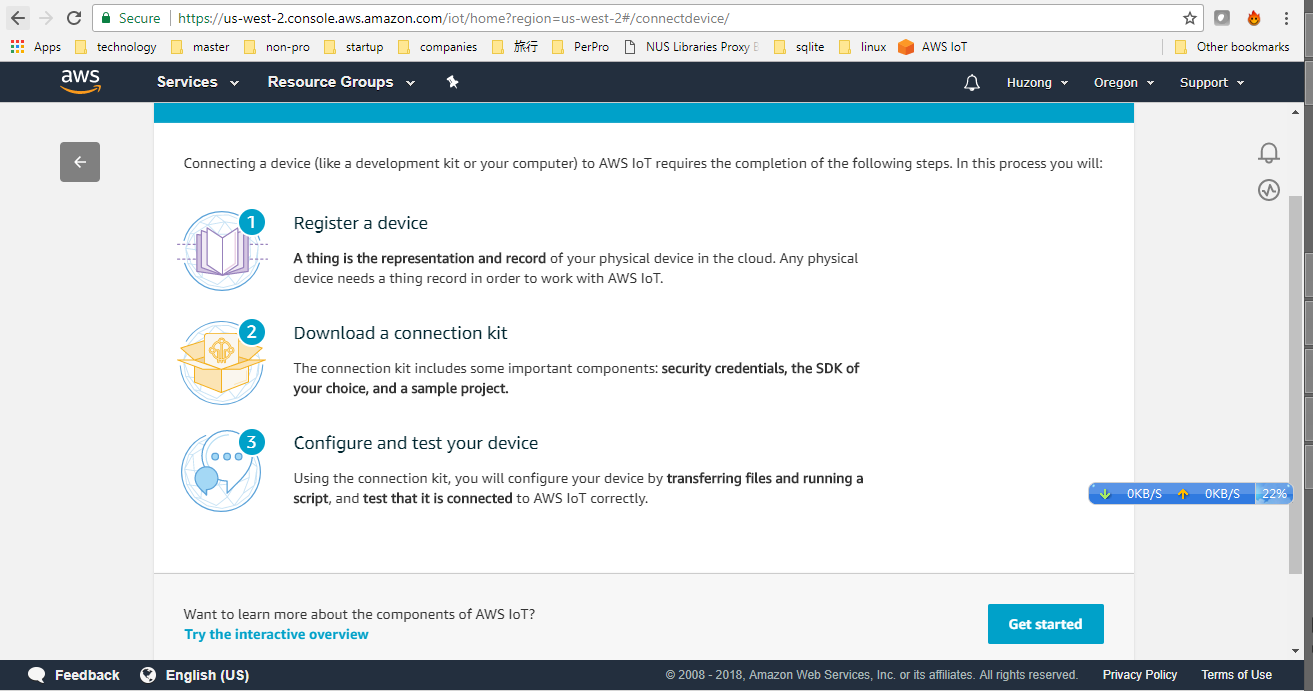


# Set up the Windows PC as the device to IOT core

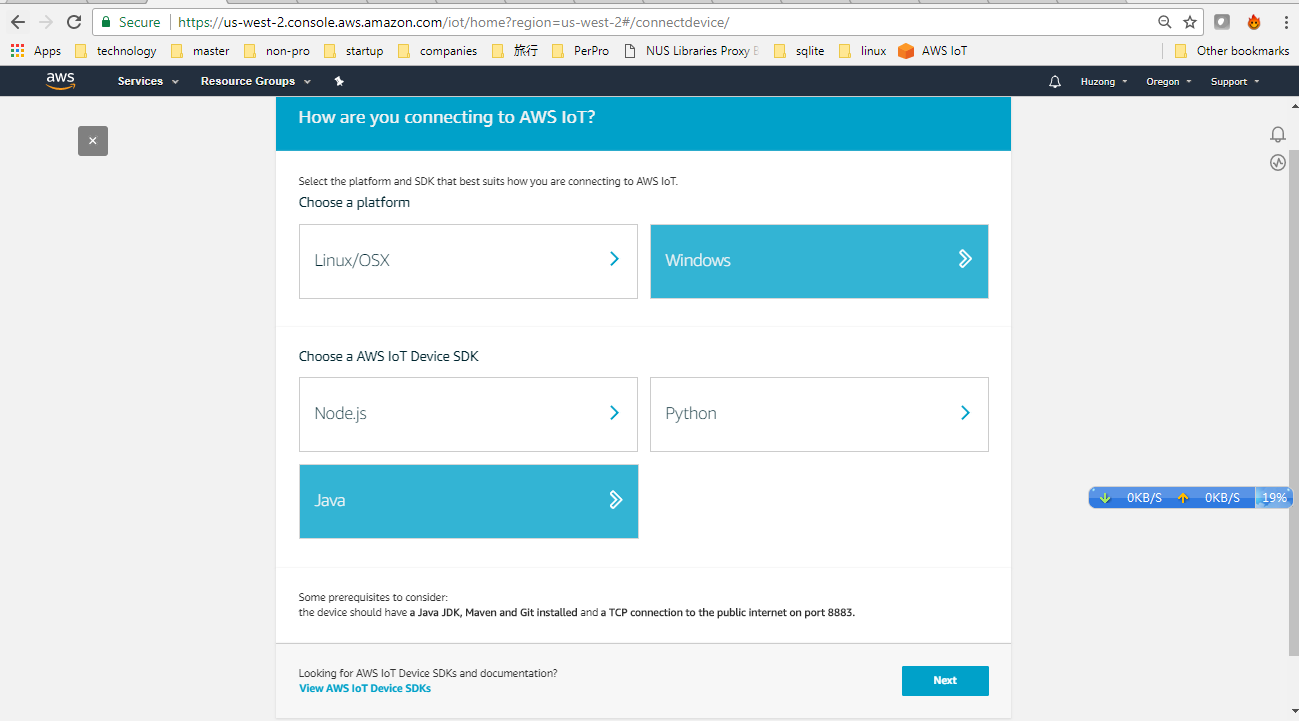
## Go to IOT core, and click the onboarding



## Click the Get Started of Configuring a Device and it will route the screen below

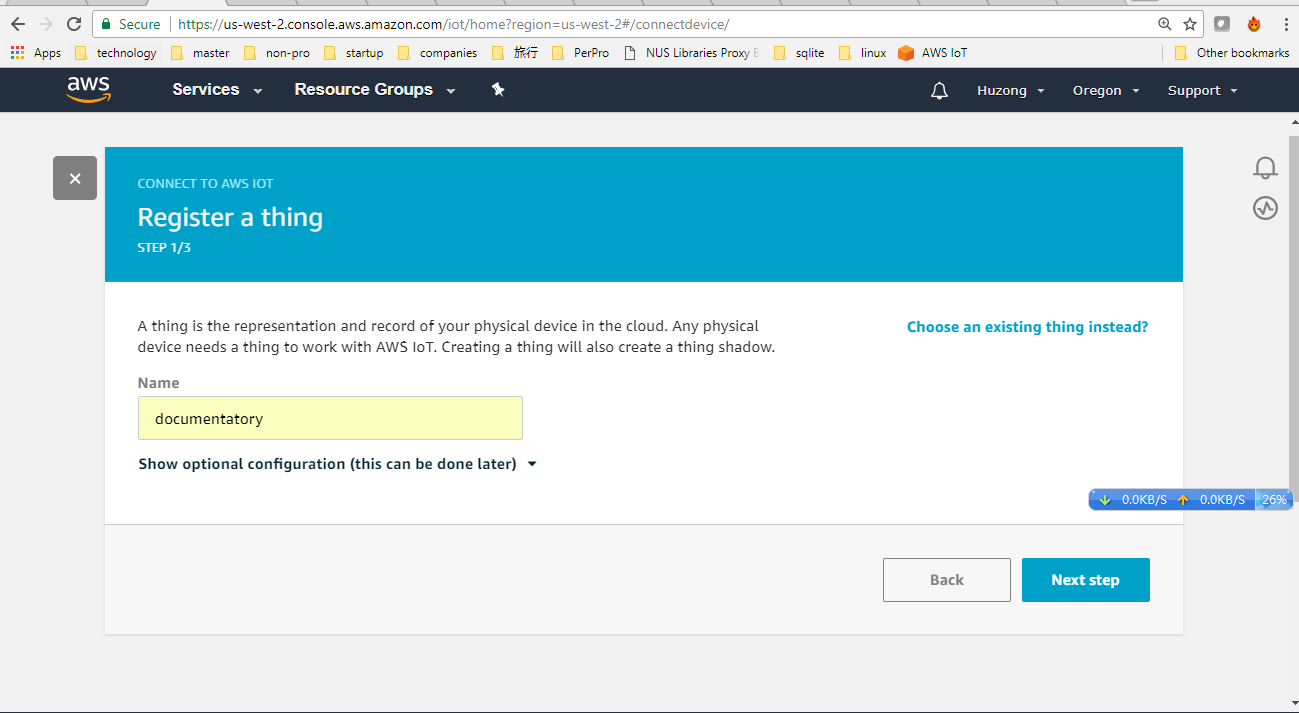


## Click the Get start and it will route the page below

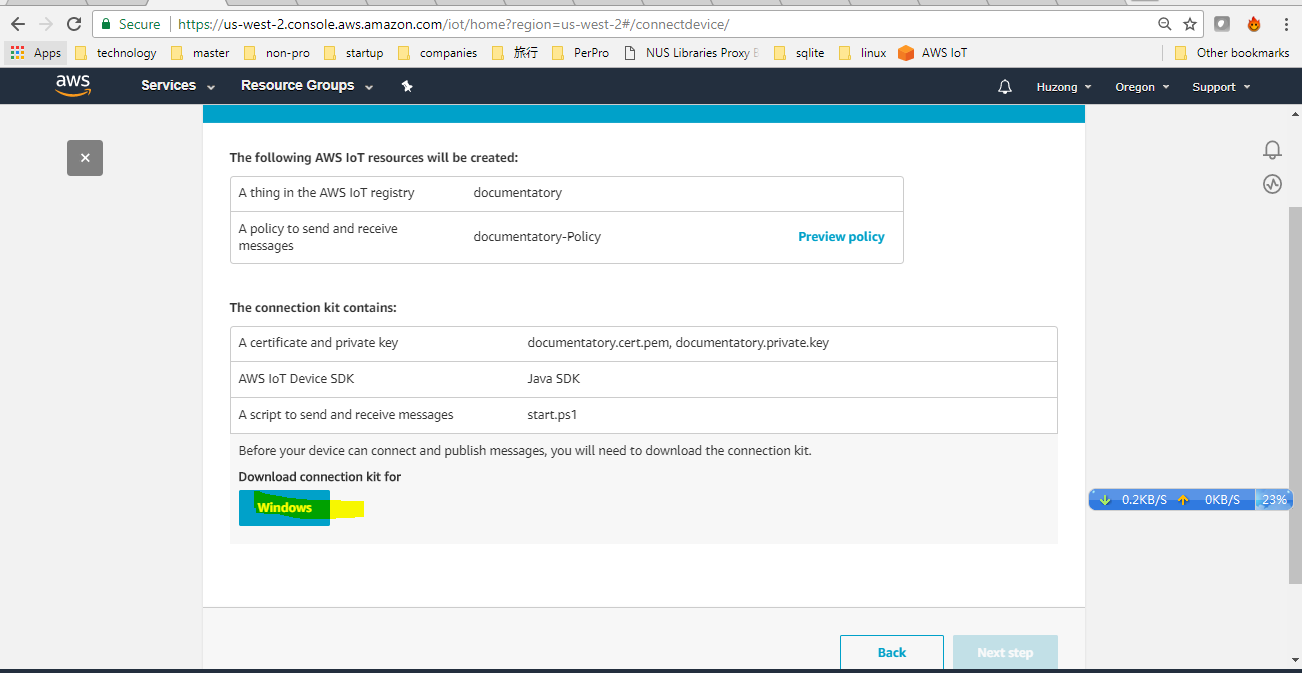


## choosing the correct OS/toolset and click the next

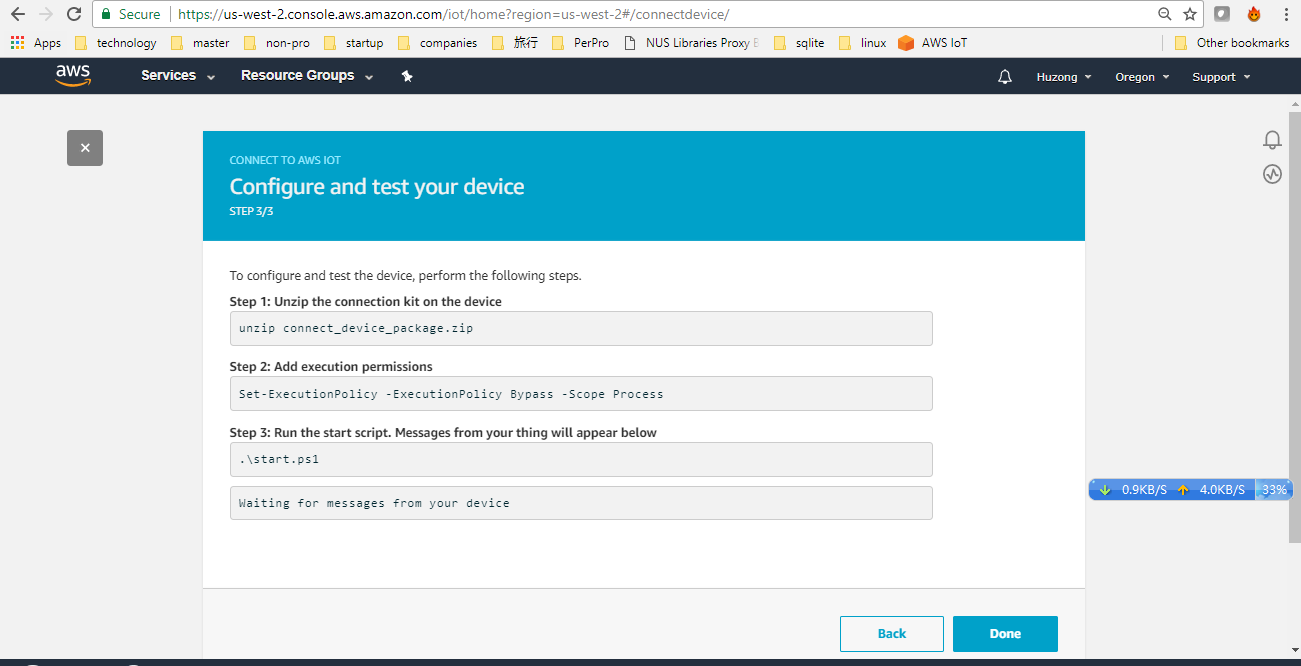
## Giving a name and click "next step"



## Click the windows to download the toolset, and next steps will become active



## After clicking next steps, it will route to the page with details text below



## To configure and test the device, perform the following steps.

**Step 1**: Unzip the connection kit on the device

unzip connect\_device\_package.zip

**Step 2**: Add execution permissions

Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process

**Step 3**: Run the start script. Messages from your thing will appear below

.\start.ps1

Waiting for messages from your device

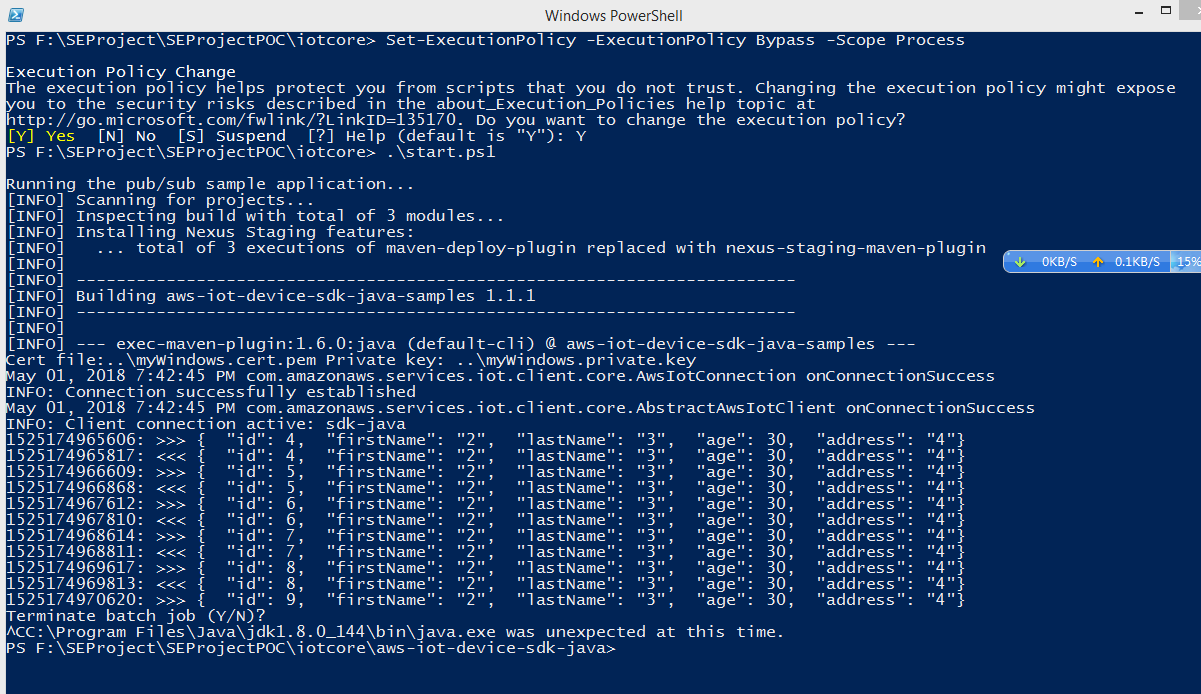
p.s. *If you encounter any problem, you may use the git repository*

*https://github.com/aws/aws-iot-device-sdk-java*

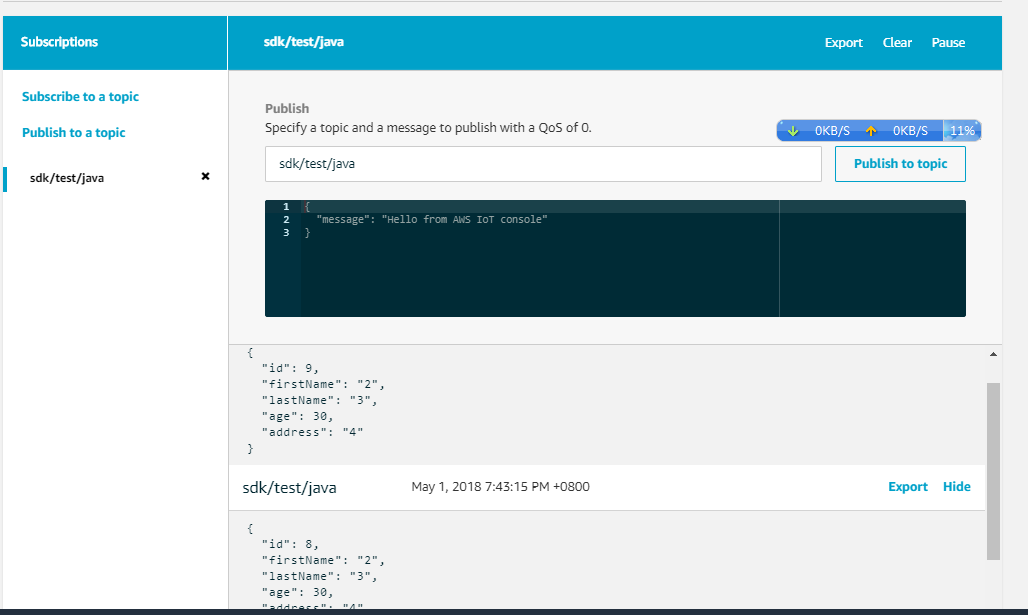
## EndToEnd UI

## Device -- IOT-- Lambda--DynamoDB

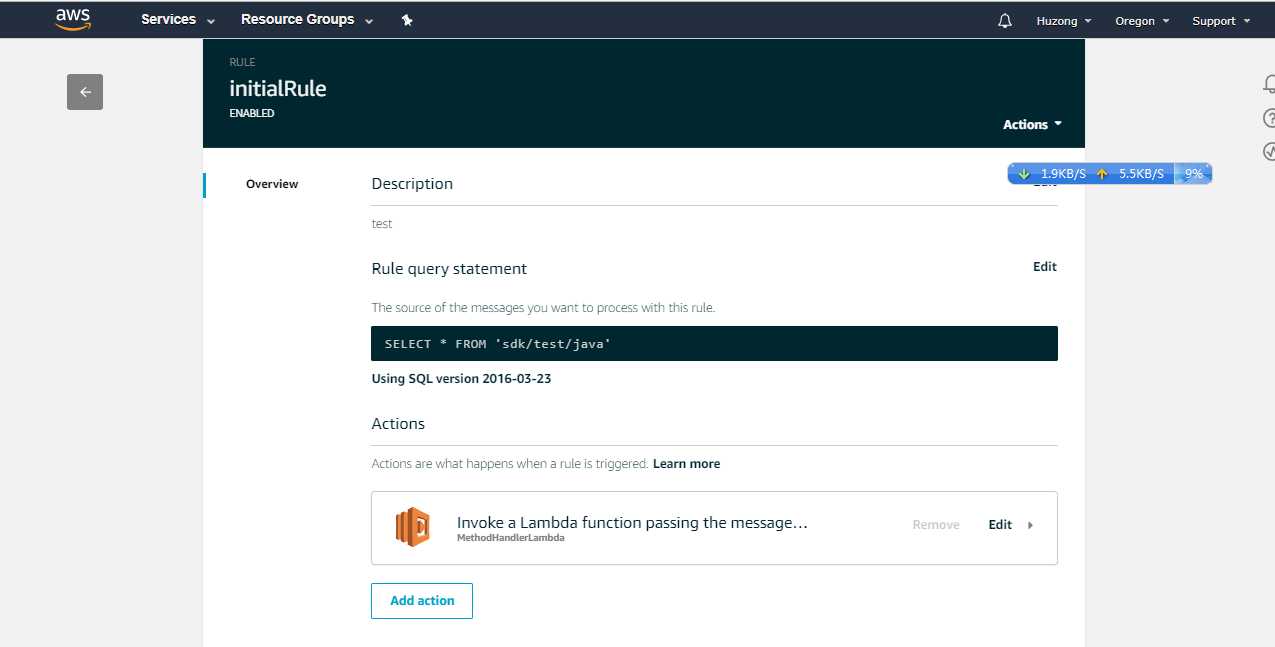
## windows acting as the device, sending data to iot core



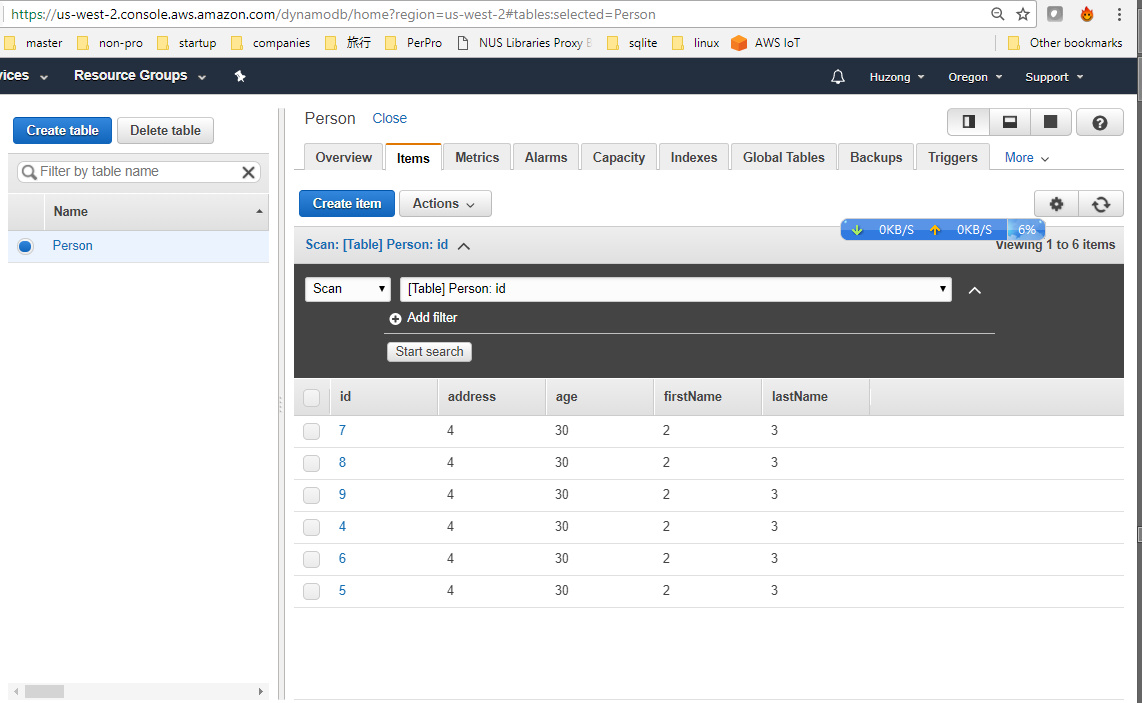
## subscribe the MQTT topic, we are able to find the msgs



## IOT set a rule, when the required msg arrived, it will trigger the lambda

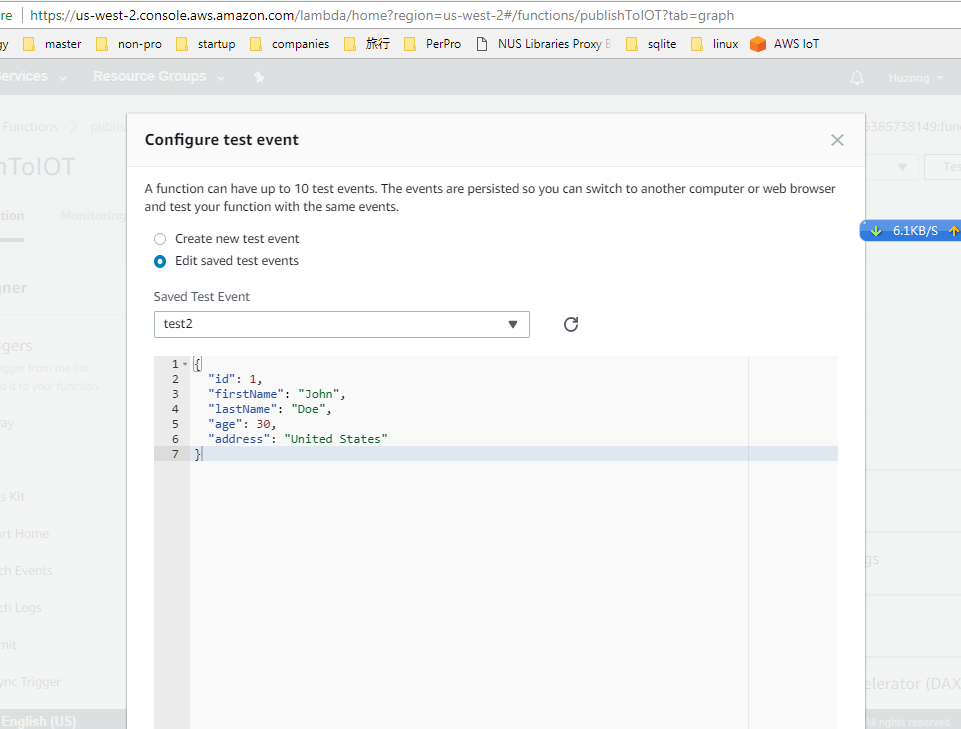


## The lambda will save the db to dynamoDB

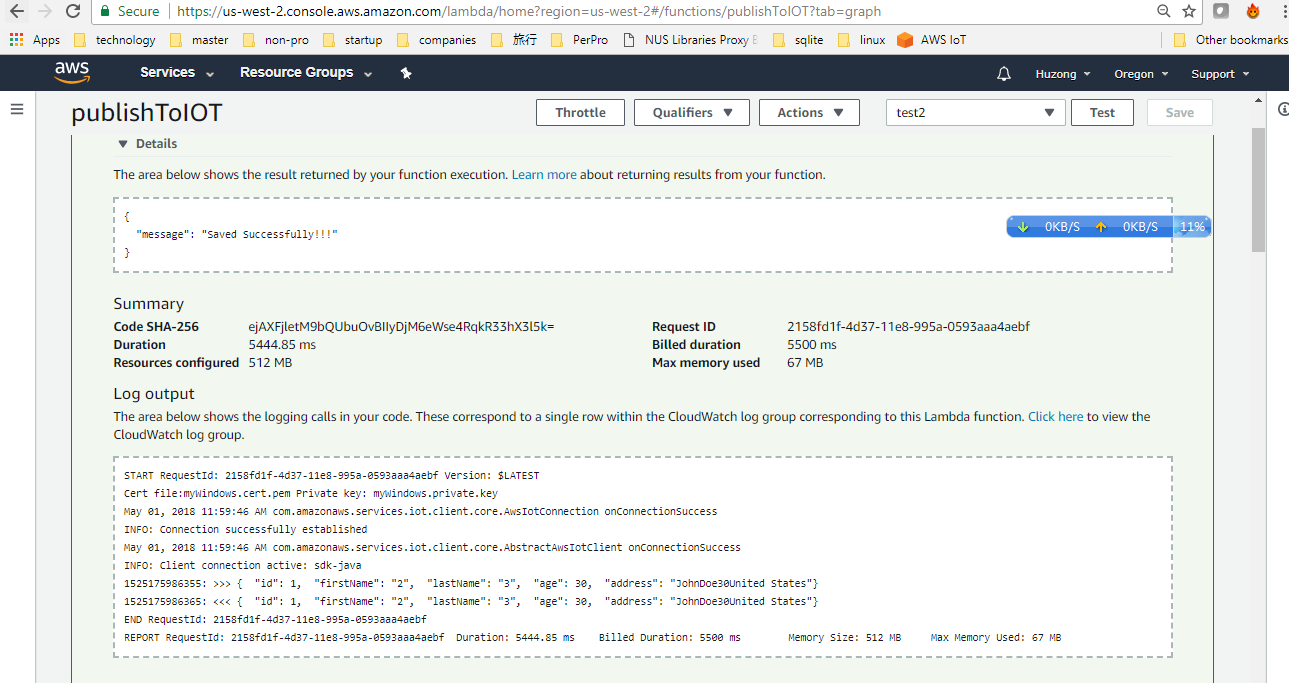


## The lambda will send data back to IOT core

## Prepare the test case



## Execute the Lambda



## Verify IOT core successfully capture the msg

Figure : The final look of the prototype

