Enable Alexa Voice Service (AVS) on Intel NUC

This tutorial explains steps to port Amazon Alexa Voice Service on Intel NUC gateway with Windriver OS, Moon Island 3.0 OS image. This tutorial assumes you have already done some basic setup on your gateway and are familiar with its operation.

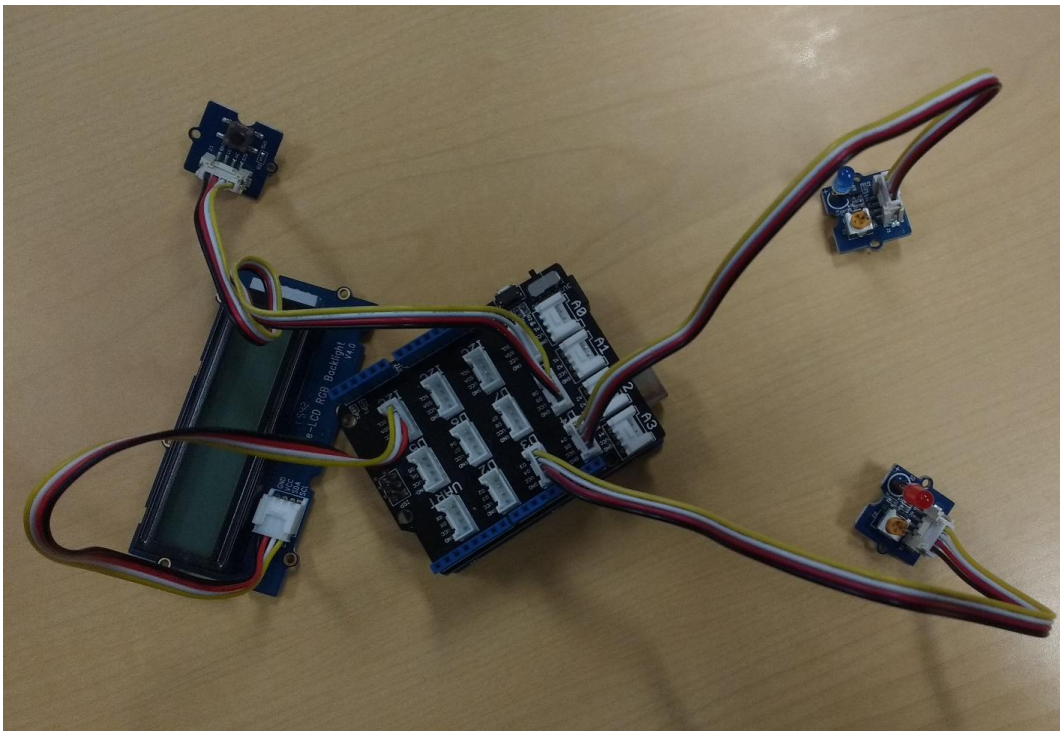
Setup Arduino 101 and Firmata:

Clone the Github repository onto your gateway by logging into the gateway and running:

git clone https://github.com/SSG-DRD-IOT/Alexa-on-Intel-NUC

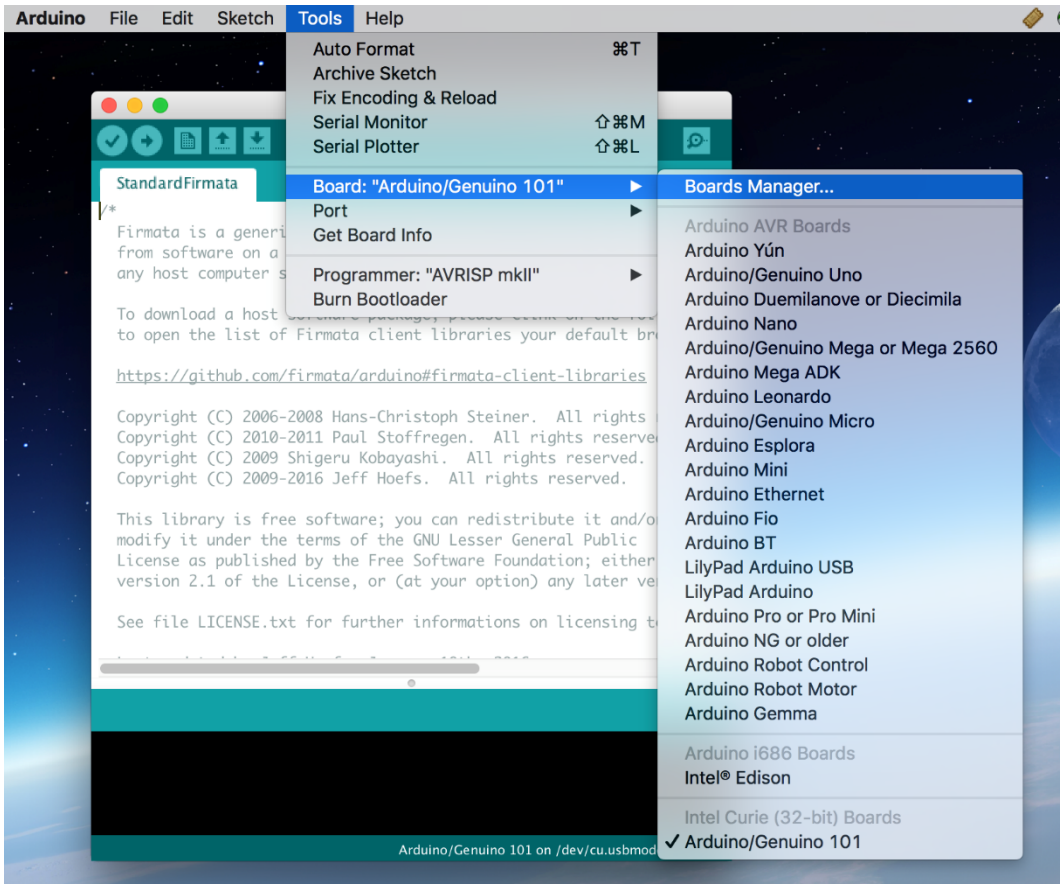
Wire up the sensors as follows:
Connect Grove button to pin D8, one LED to pin D3 and other one to Pin D4 as shown in the following picture using the cables from the grove starter

sensor kit box.

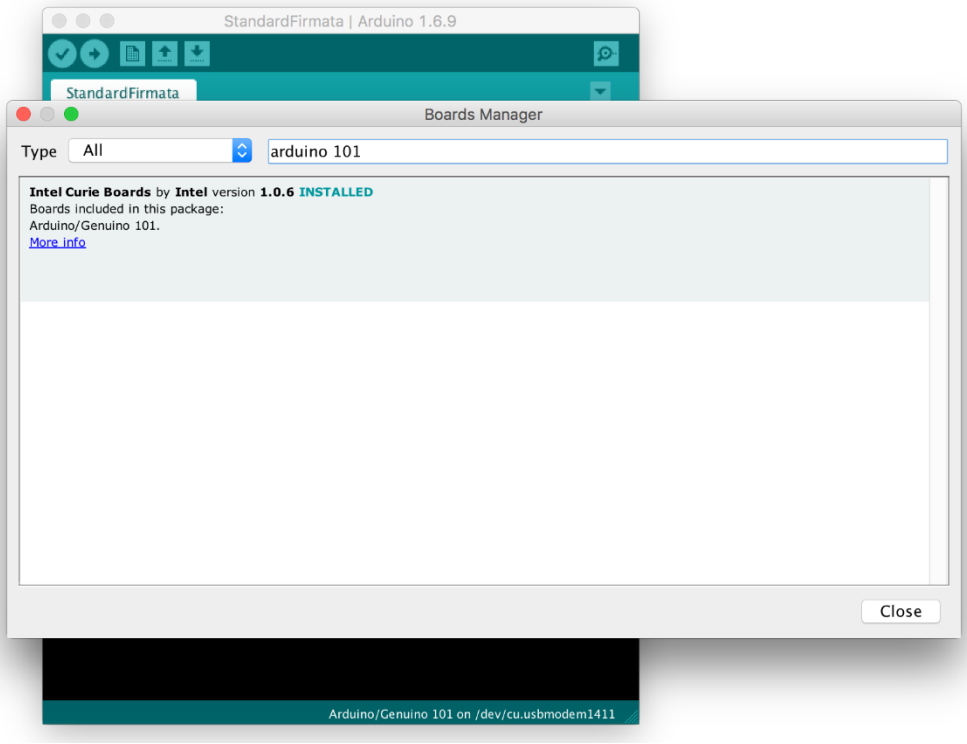


- Set Up Firmata on Arduino 101:

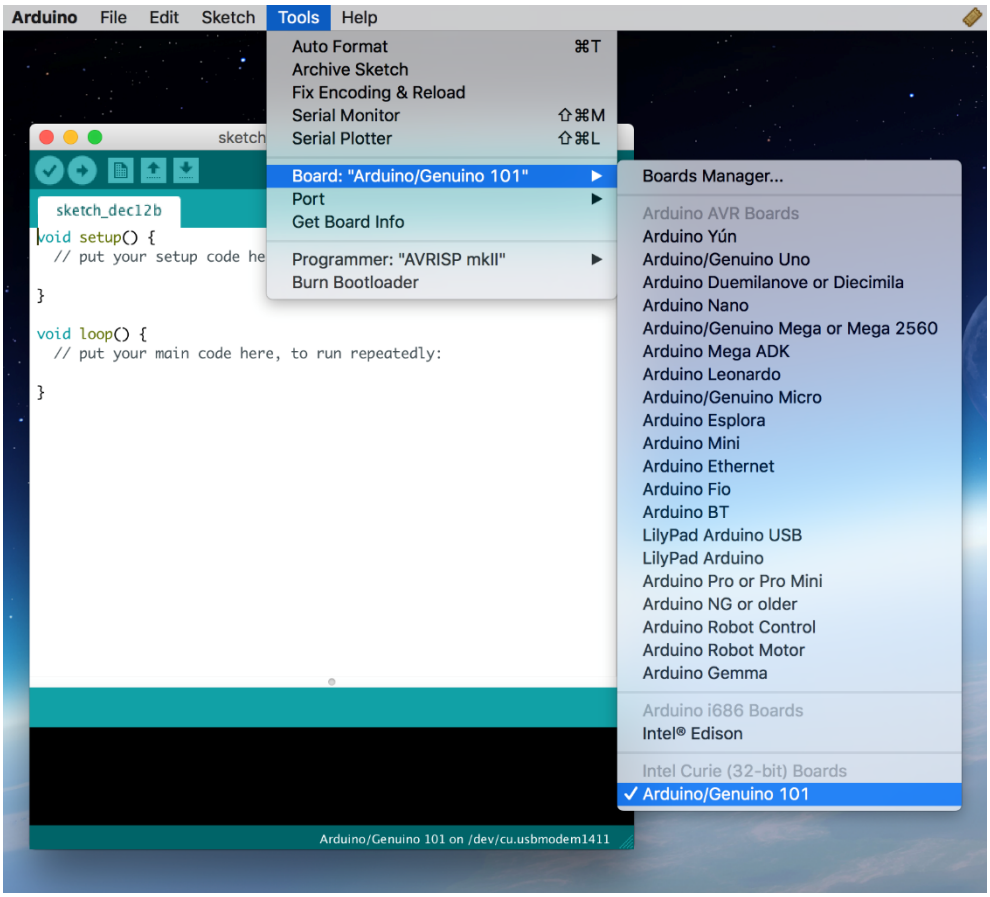
a. Download Arduino IDE: https://www.arduino.cc/en/Main/Software
b. Connect Arduino 101 to host computer via USB
c. Launch Arduino IDE
d. Open boards manager:



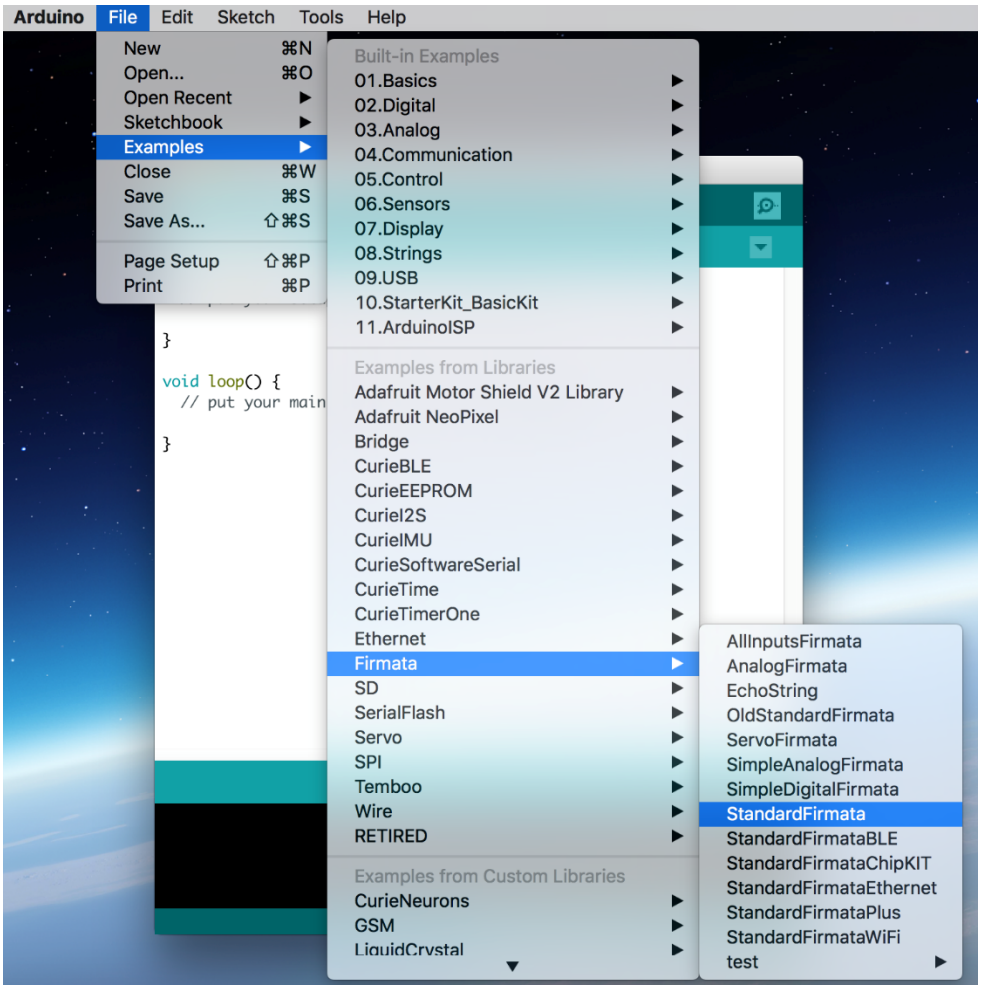
e. Search for “Arduino 101” and install the board definition:



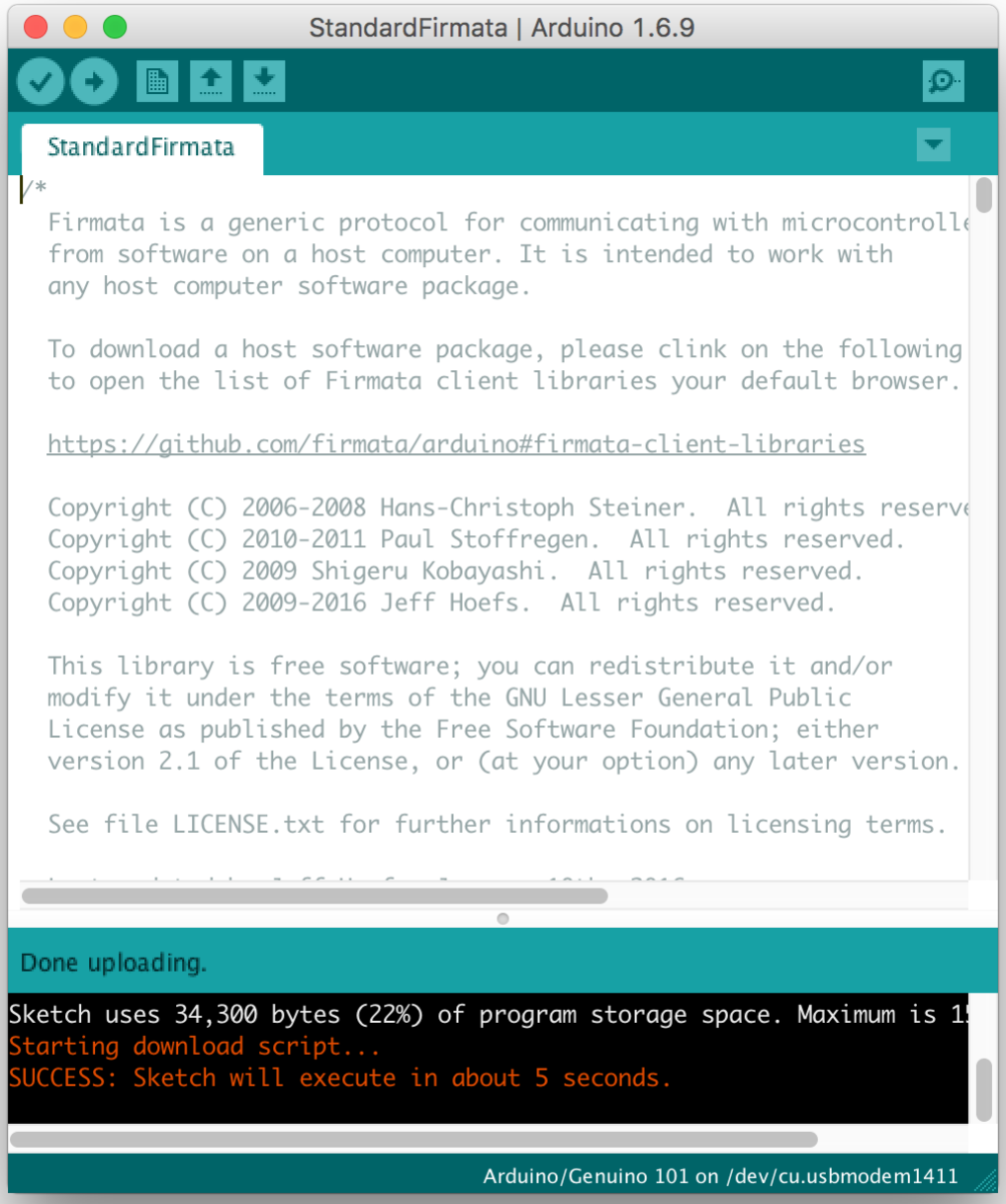
f. Select Arduino/Genuino 101 from the Boards menu and the correct port. If you are unsure which port to use disconnect the Arduino 101 and see which port disappears.



g. Load StandardFirmata Sketch:



h. Click Upload and wait for “Sketch will execute about 5 seconds”


StandardFirmata | Arduino 1.6.9
StandardFirmata
/*
Firmata is a generic protocol for communicating with microcontrolle
from software on a host computer. It is intended to work with
any host computer software package.
To download a host software package, please clink on the following
to open the list of Firmata client libraries your default browser.
https://github.com/firmata/arduino#firmata-client-libraries
Copyright (C) 2006-2008 Hans-Christoph Steiner. All rights reserve
Copyright (C) 2010-2011 Paul Stoffregen. All rights reserved.
Copyright (C) 2009 Shigeru Kobayashi. All rights reserved.
Copyright (C) 2009-2016 Jeff Hoefs. All rights reserved.
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
See file LICENSE.txt for further informations on licensing terms.
Done uploading.
Sketch uses 34,300 bytes (22%) of program storage space. Maximum is 1
Starting download script...
SUCCESS: Sketch will execute in about 5 seconds.
Arduino/Genuino 101 on /dev/cu.usbmodem1411

Setup Gateway as Echo:

Run the setup script, setup.sh, from github repository https://github.com/SSG-DRD-IOT/Alexa-on-Intel-NUC

Reboot the Gateway

Generate SSL Key for Intel NUC
http://docs.cherrypy.org/en/3.3.0/progguide/security.html

Run these commands:

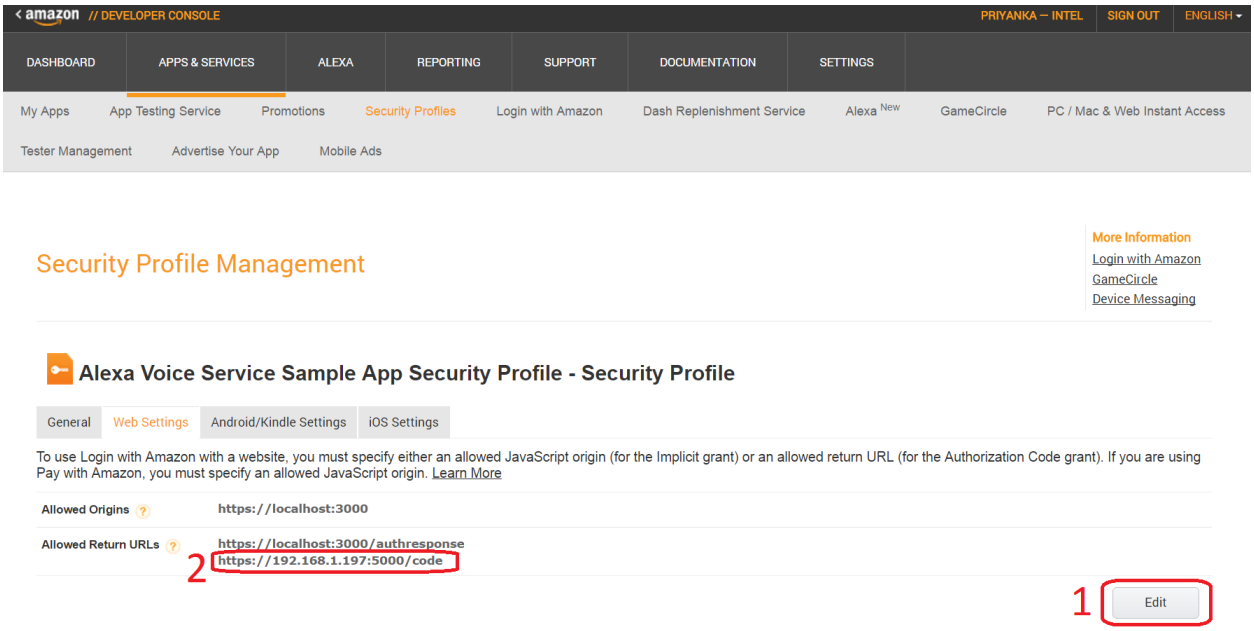
```
$openssl genrsa -out /root/privkey.pem 2048
$openssl req -new -x509 -days 365 -key /root/privkey.pem -out /root/cert.pem
```

Follow this link to create AWS account and security profile
https://github.com/alexa/alexa-avs-sample-app/wiki/Linux

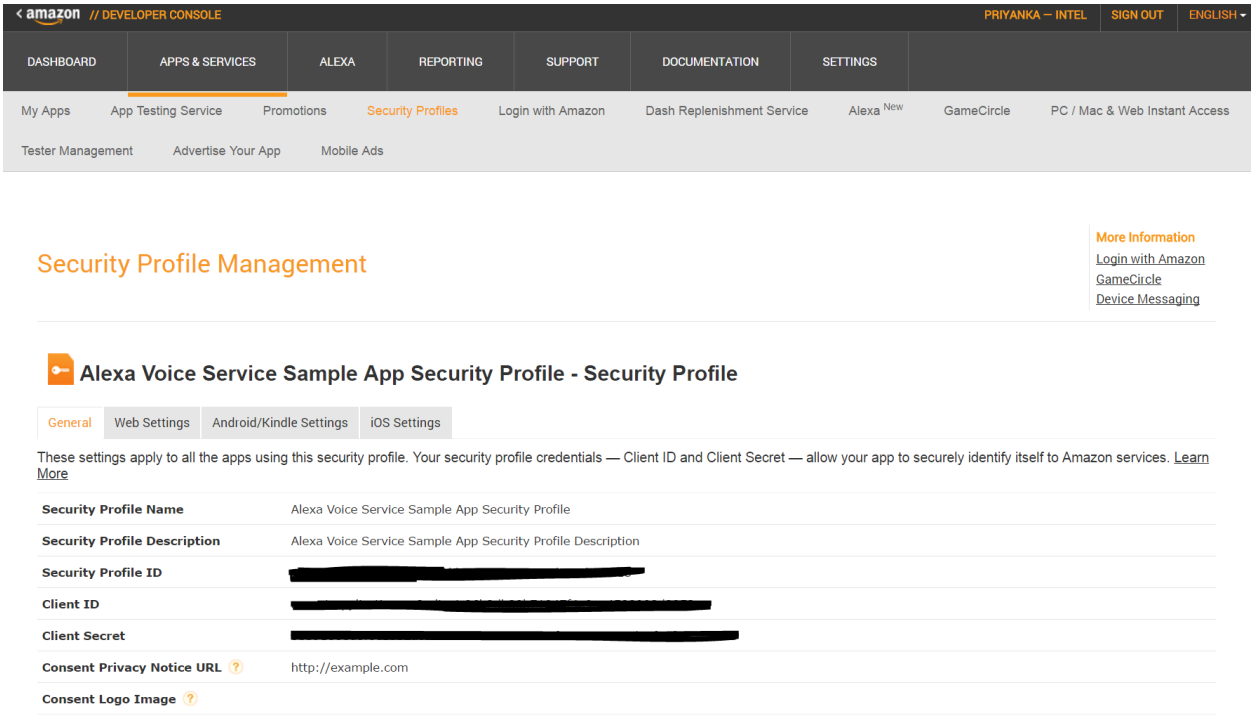
Security profile setup for Intel NUC on Amazon Developer console
Login to Amazon Developer Portal - developer.amazon.com

APPS & SERVICES -> Security Profile

Click Web setting tab



Click Edit button on Web settings tab.
Get IP address of your NUC using ifconfig command.
Add the URL in the form https://ip_address_for_NUC:5000/code
e.g. https://192.168.1.197:5000/code



Important!: Edit creds.py and add the information in the general security tab to the file.

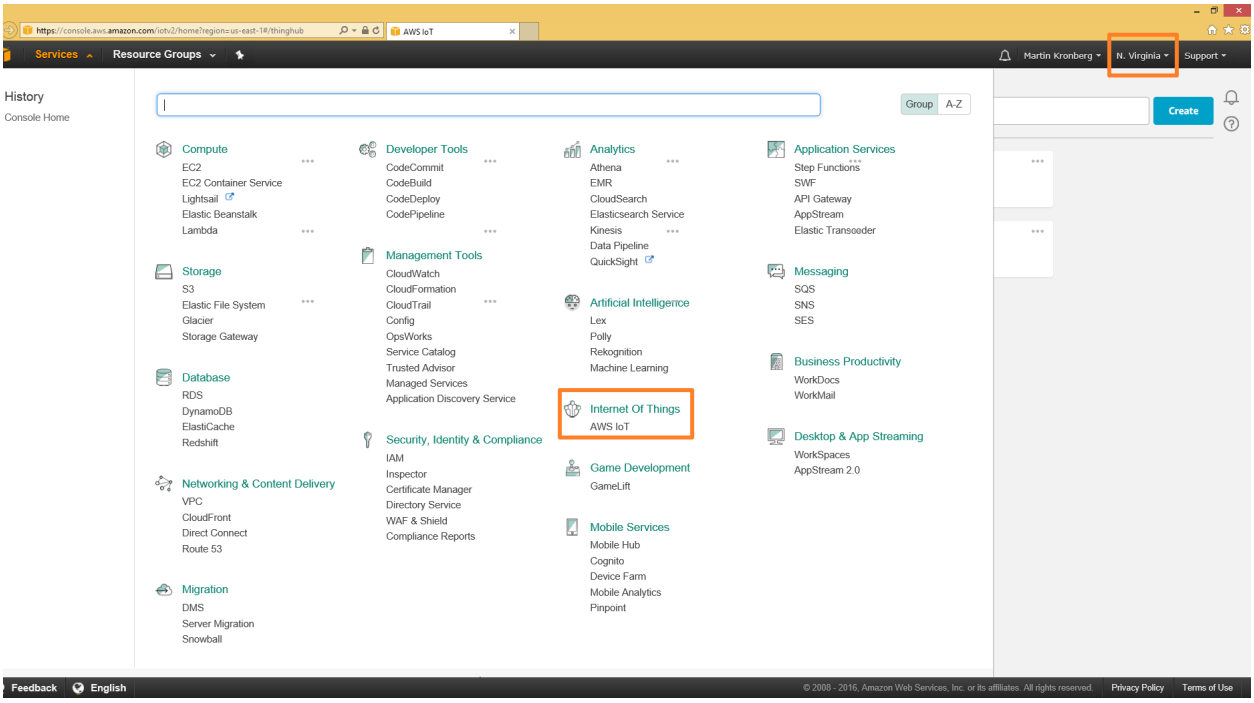
Create Custom Skills For The Gateway to control sensors:

Create an AWS IoT Device:

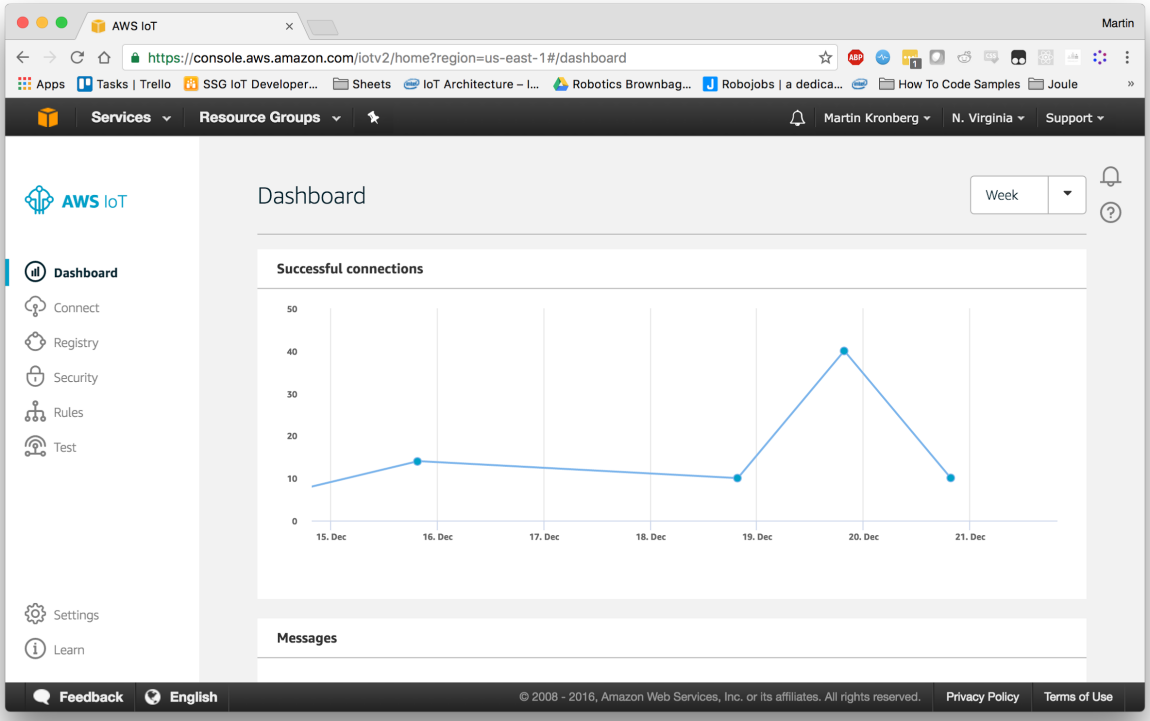
In order for the gateway to communicate with the AWS cloud we will create an AWS IoT Device to act as a bridge. Alexa voice services will then push updates to the device shadow (a virtual persistent state) and the gateway will get updates from the device shadow.

Create an AWS account: https://console.aws.amazon.com

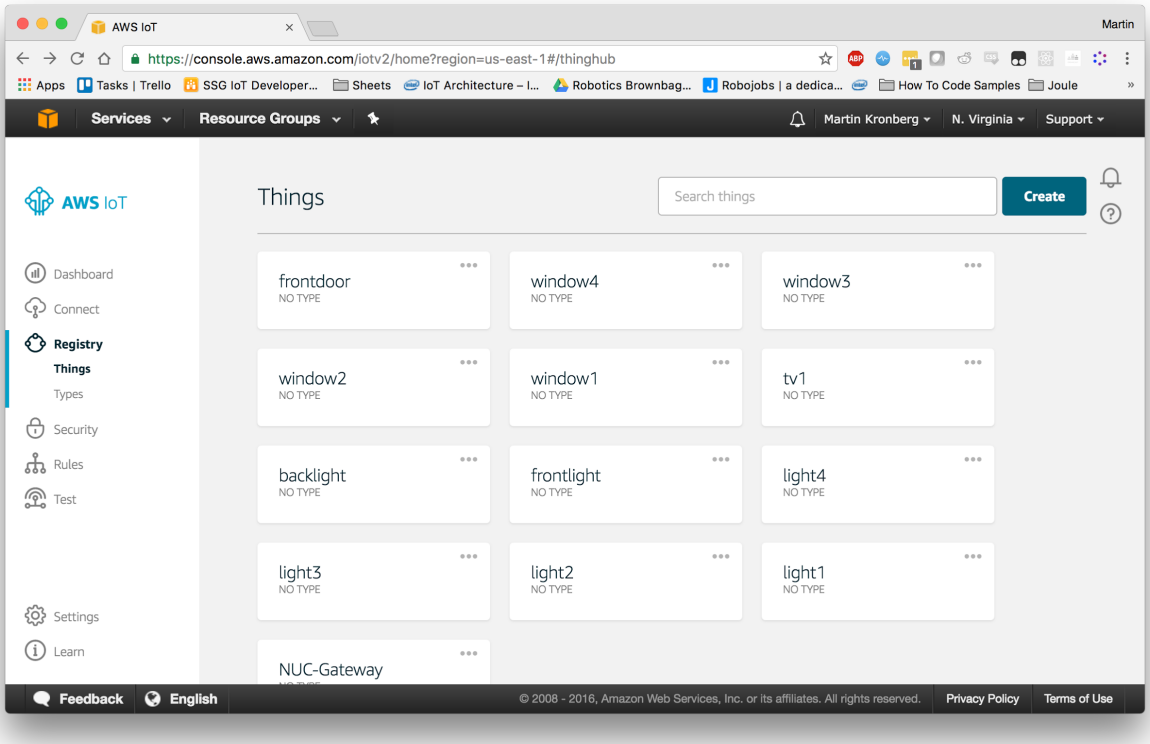
From Services select AWS IoT:
Make sure that your region is selected as N. Virginia (us-east-1). The AWS services are not yet online for all regions.



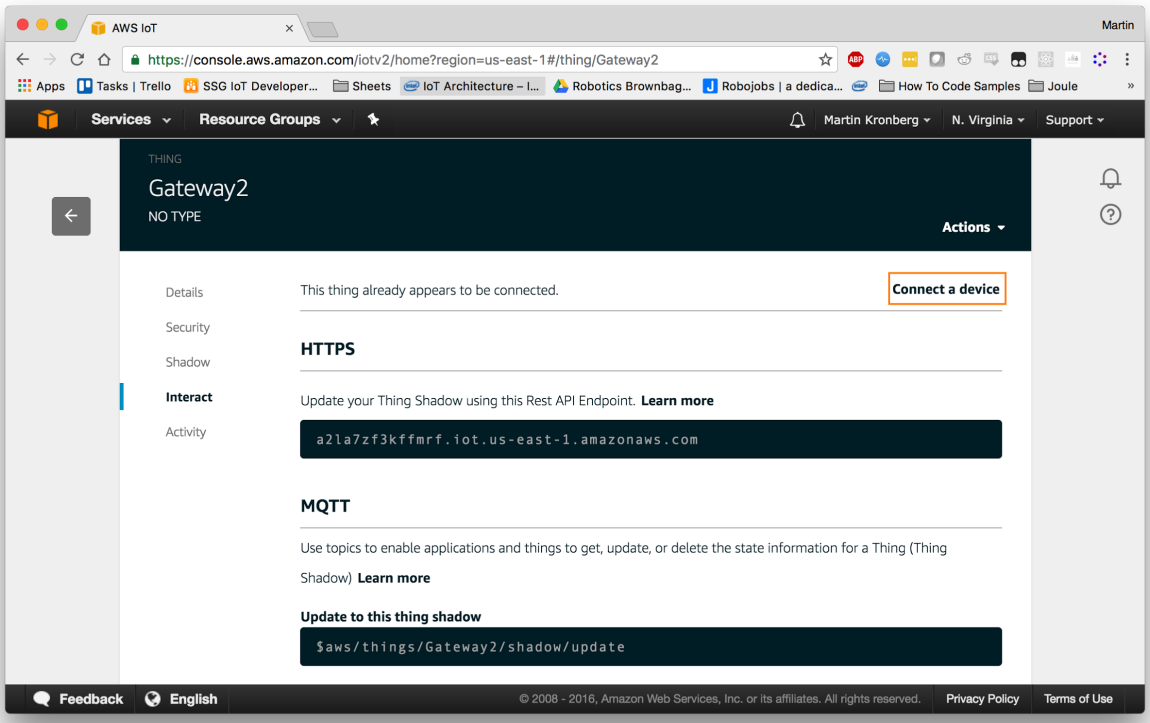
The dashboard should look like this:


AWS IoT
Martin
https://console.aws.amazon.com/iotv2/home?region=us-east-1#/dashboard
Apps
Tasks | Trello
SSG IoT Developer...
Sheets
IoT Architecture – I...
Robotics Brownbag...
Robojobs | a dedica...
How To Code Samples
Joule
Services
Resource Groups
Martin Kronberg
N. Virginia
Support
AWS IoT
Dashboard
Connect
Registry
Security
Rules
Test
Settings
Learn
Dashboard
Week
Successful connections
50
40
30
20
10
0
15. Dec
16. Dec
17. Dec
18. Dec
19. Dec
20. Dec
21. Dec
Messages
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

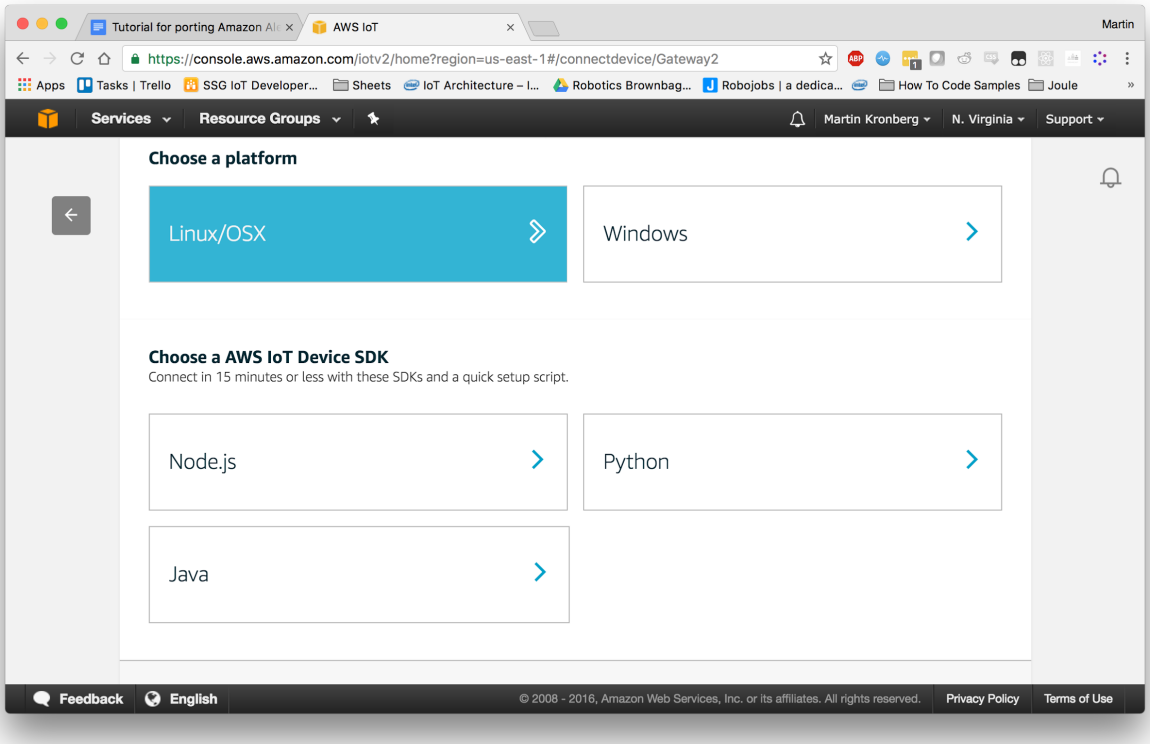
Go to: Registry -> Things and click “Create” to the right of the search bar. Follow the screens to create AWS IoT device.



On the device page navigate to Interact and click on “Connect a Device”



Choose your platform – Linux/OSX or Windows, and choose Python.



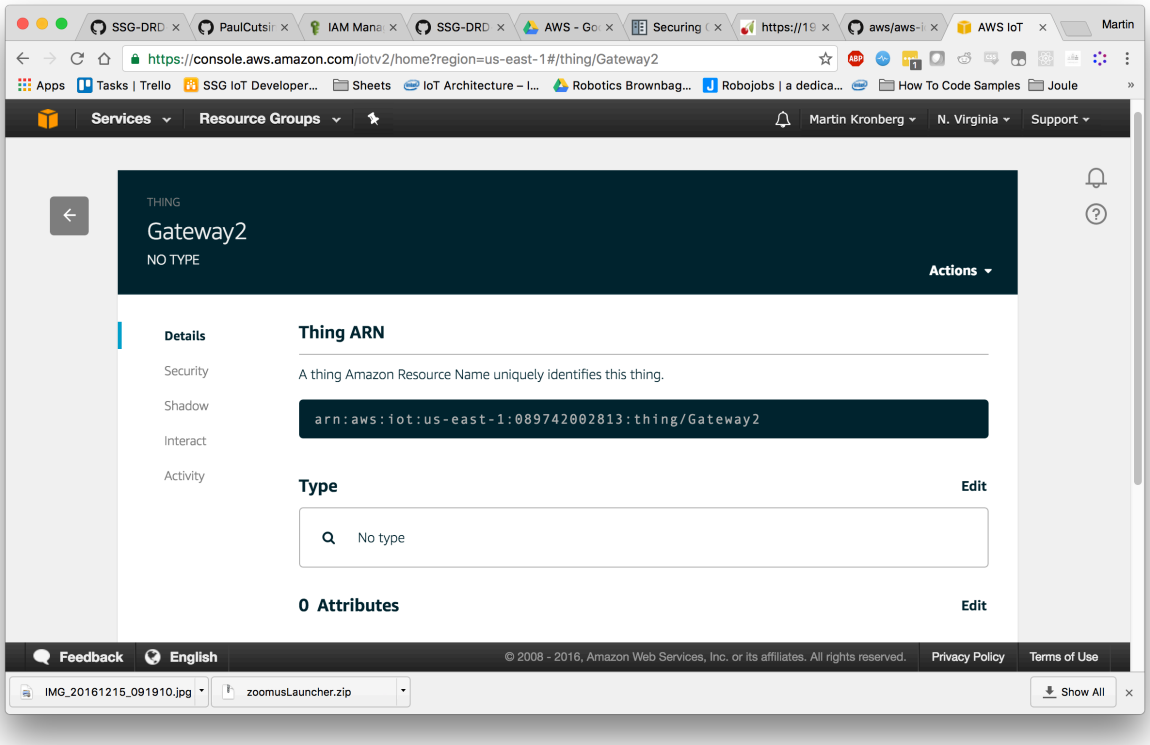
Click “Getting Started” and download the connection kit for your selected OS. This package will contain a private key and certificate which we will use to connect the gateway to the AWS IoT service. It also contains a start.sh script which will run a basic connection test, install the SDK, and download the root-CA.crt. You can ignore this file.

Unzip the file into the main repository folder, alongside main.py

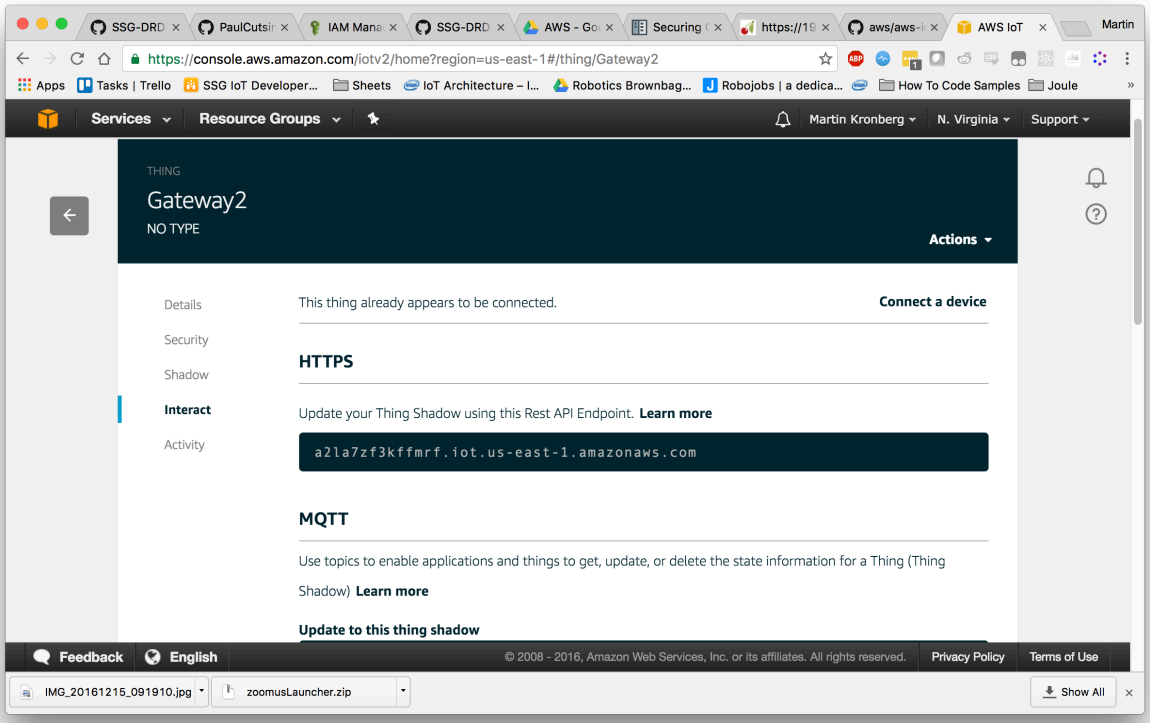
Edit main.py:

Main.py contains code for both sending voice commands to Alexa and connecting to AWS IoT. Now that you have created a device you can add its connection details to main.py.

Replace AWS_IOT_ARN with the Thing arn from the Details section of the AWS IoT device console:



Replace REST_API_ENDPOINT with the Rest API Endpoint under the HTTPS section of the Interact section of the AWS IoT Console:

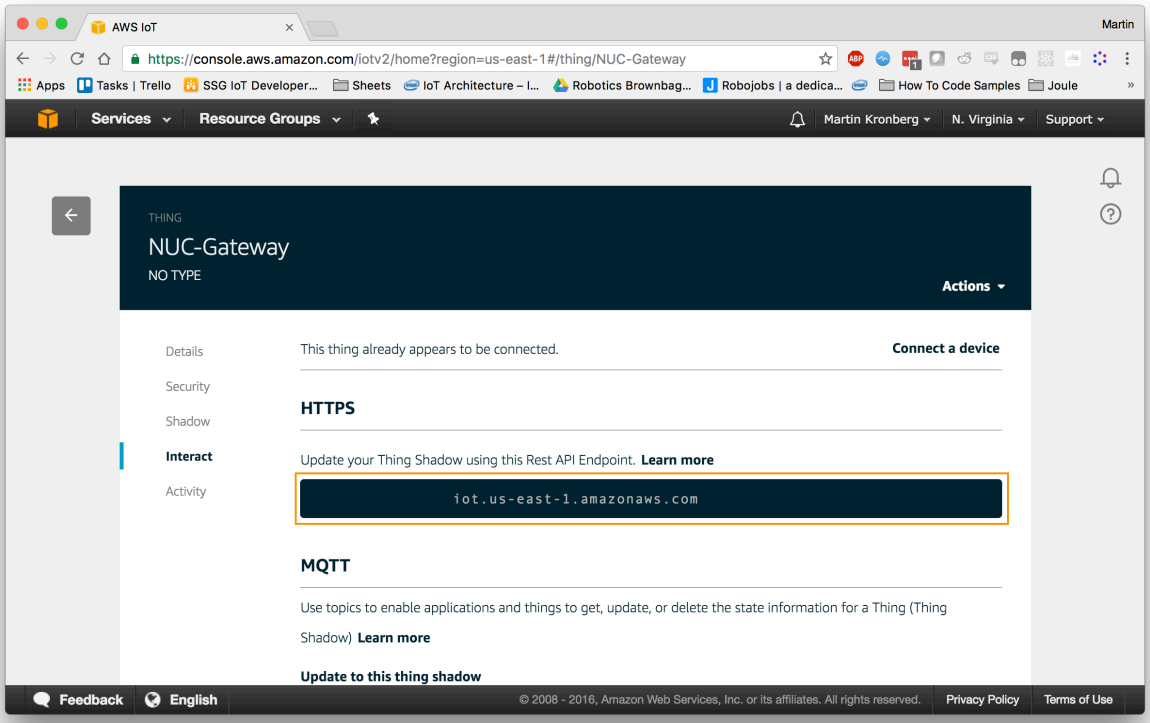


Create a Lambda Function:

The lambda function is a function that runs in a virtual run time environment in AWS. We will use it to do two things: set up a handler for requests coming from Alexa Voice Services, and setup a connection to AWS IoT to update the Gateway device shadow.

The Github repository contains /lambda/index.js which you will have to edit in order to connect to your AWS IoT Device.

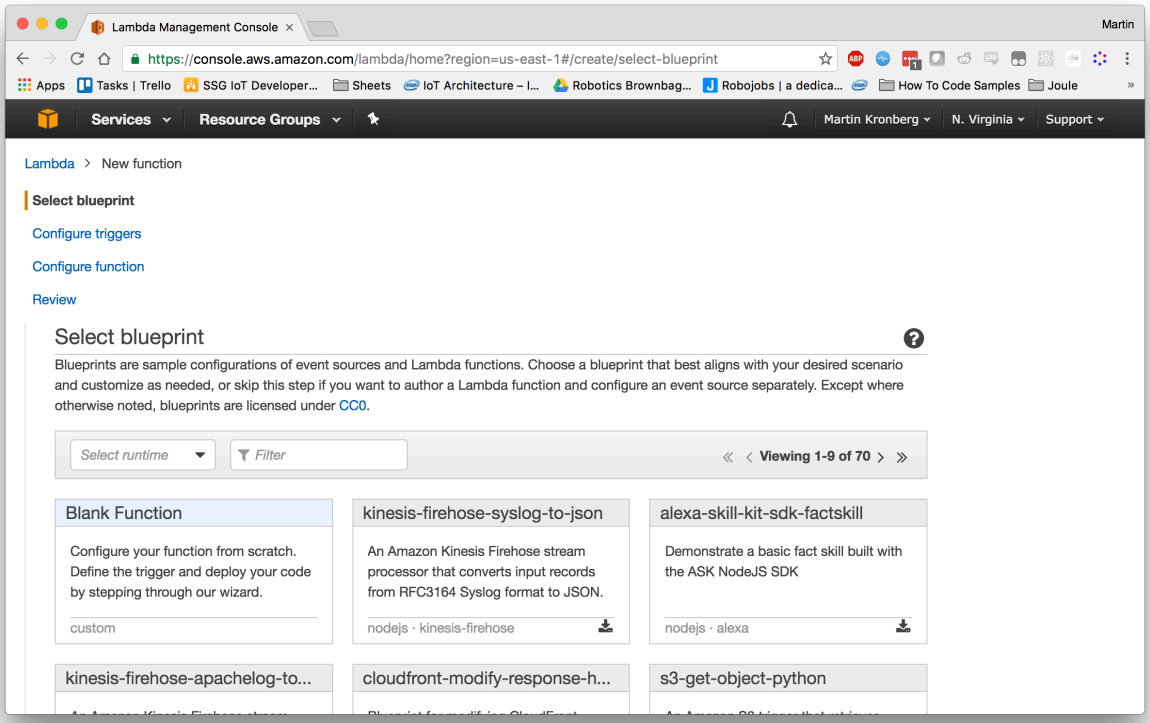
On line 4 of index.js replace REST_API_ENDPOINT with the endpoint of your IoT Device. The endpoint can be found in the AWS IoT device console under **Interact**



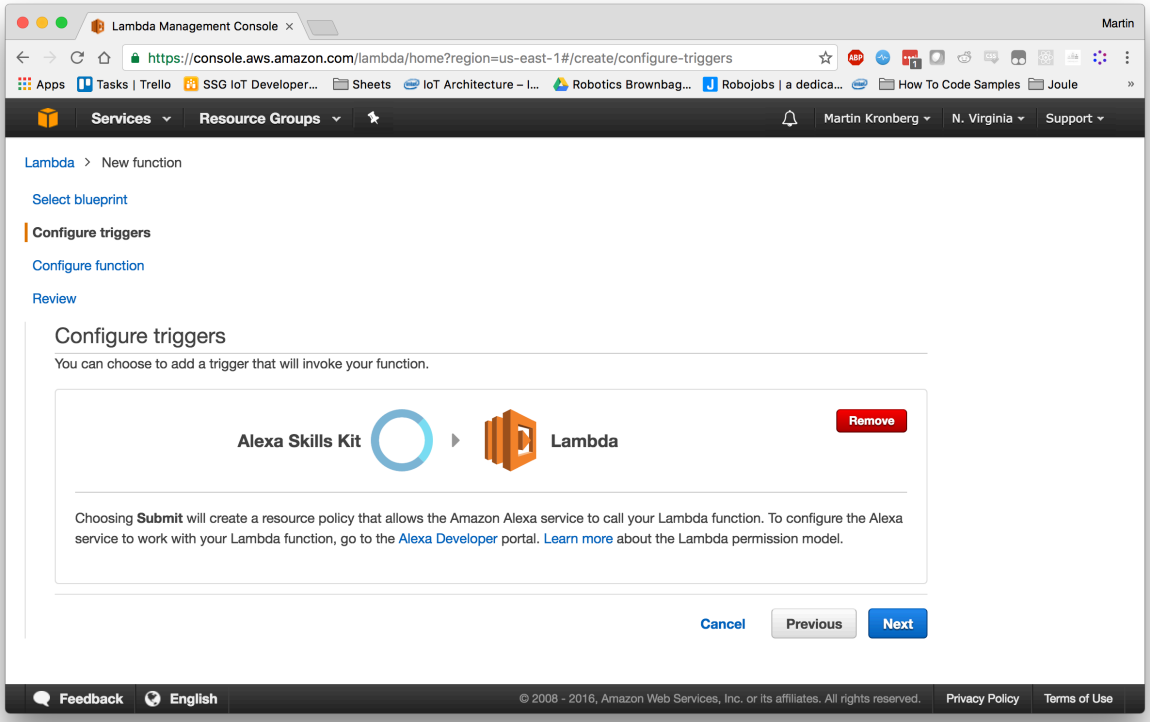
Once you have edited index.js make a zip file containing index.js and /lambda/node_modules. You will upload this to Lambda later on.

Navigate to the AWS Lambda service: https://console.aws.amazon.com/lambda/

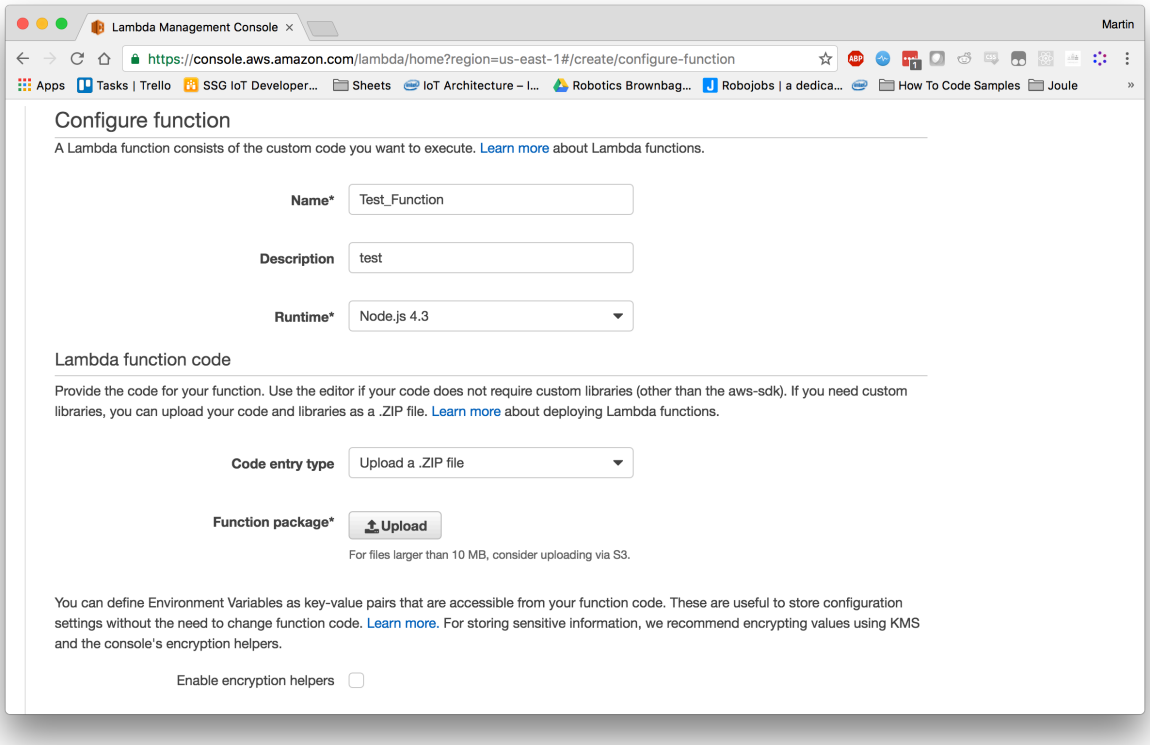
Click ‘Create Function’ use the Blank Function blueprint:



Select “Alexa Skills Kit” as the trigger:



Give your function a name and description and choose “Upload a .ZIP file”

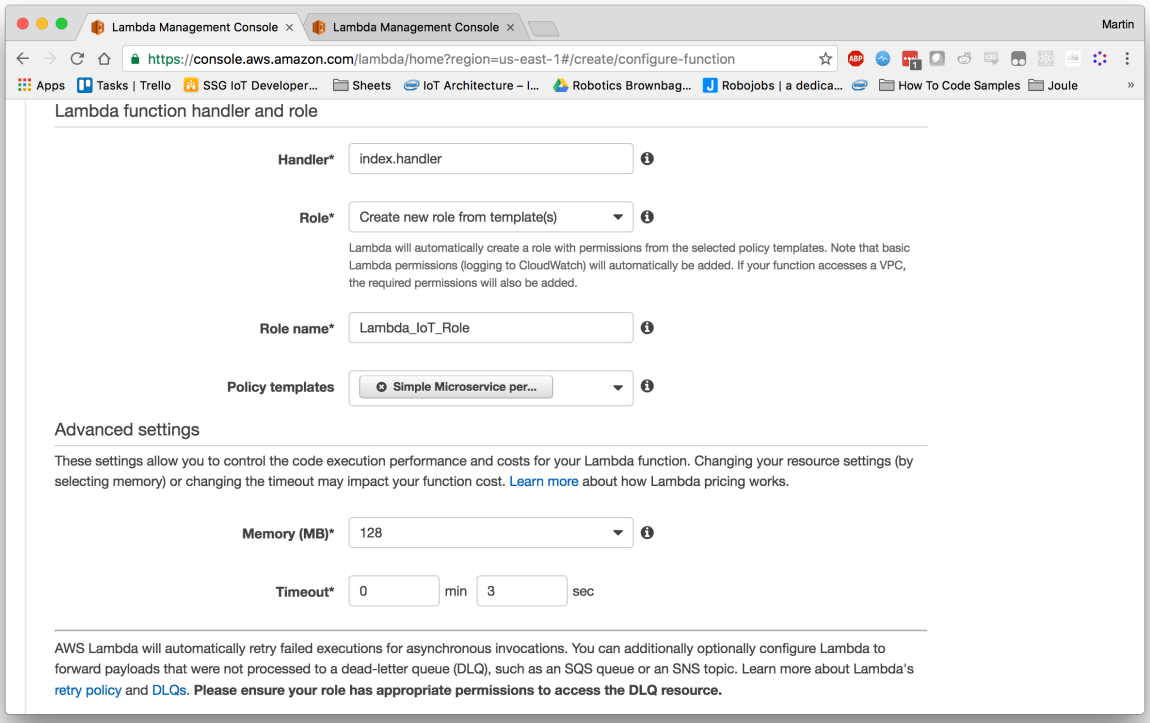


Click Upload and select the zip file you made earlier.

Enter index.handler for the Handler – this is the default handler for Lambda

Choose “Create new role form template”

Select ”Simple Microservice Permissions” from “Policy Templates”



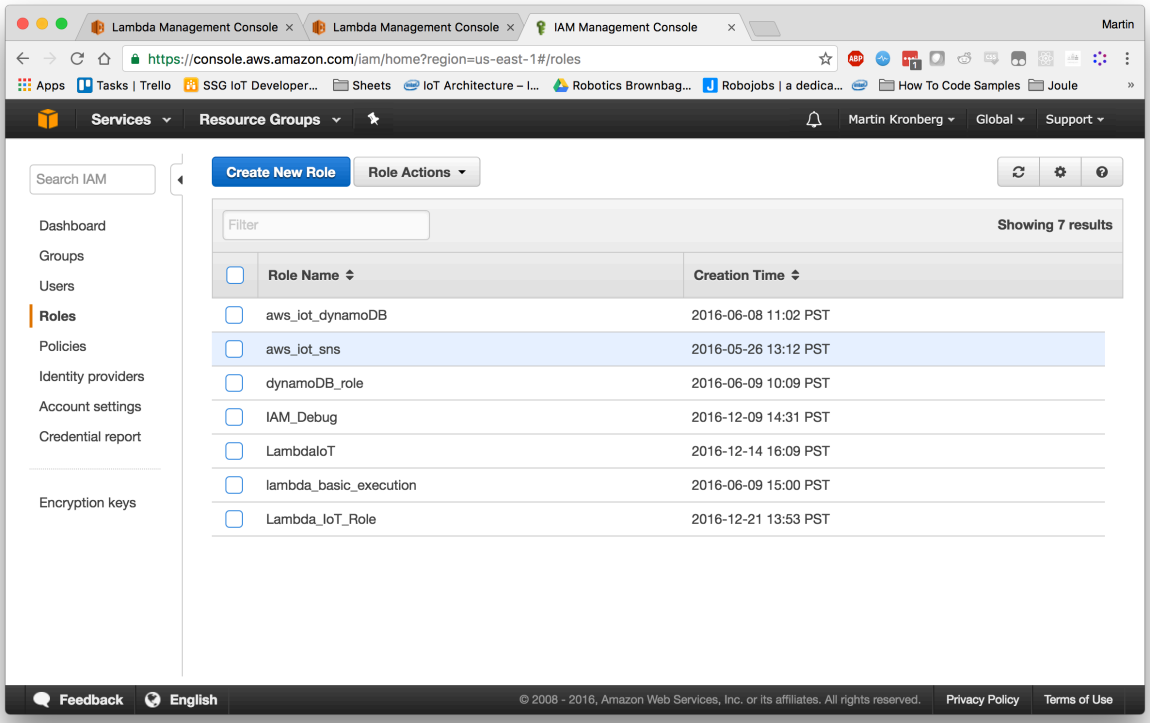
Leave everything else as default and click next.

Review the function on the next page and click “Create Function”

Configure IAM access policy:

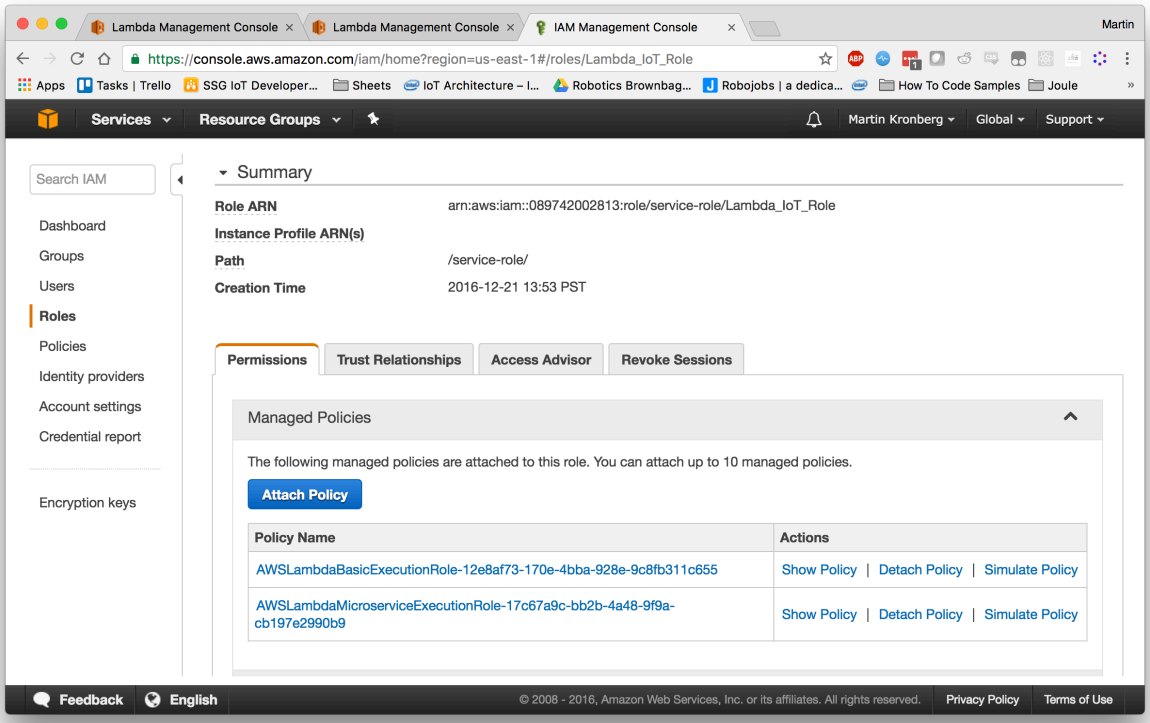
Navigate to AWS IAM console: https://console.aws.amazon.com/iam

Navigate to the Roles list form the left side menu:

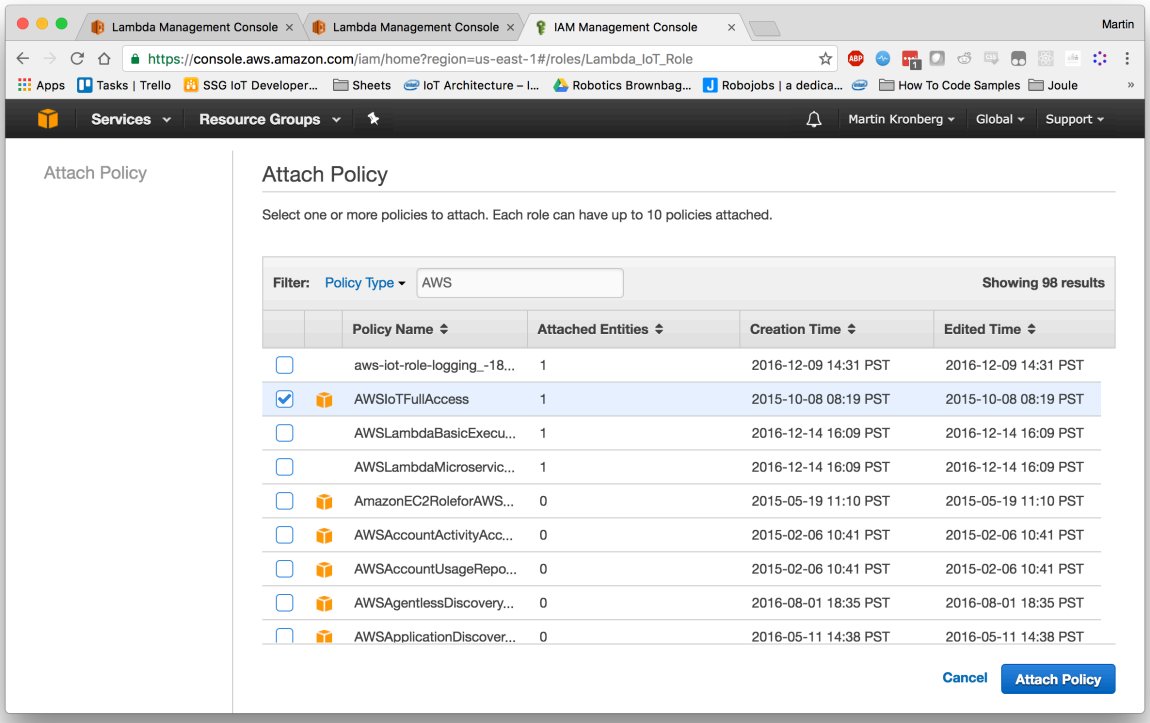


Select the role policy you created for your Lambda function

Select “Attach Policy”



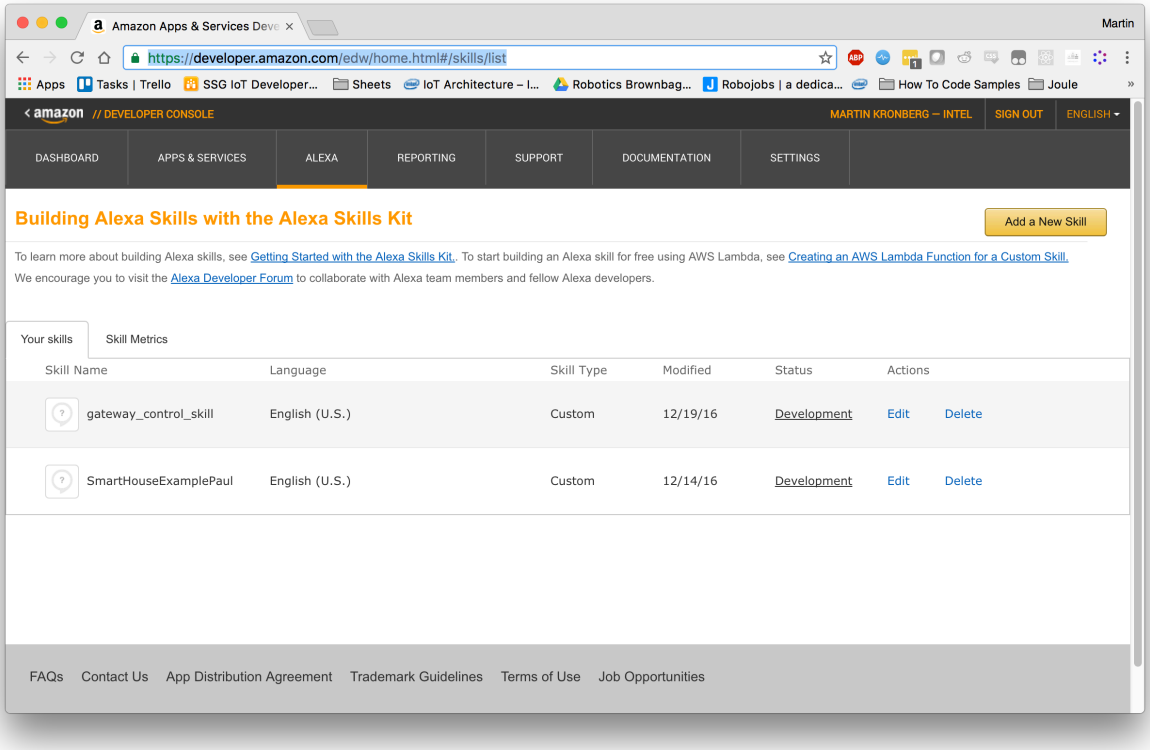
Select “AWSIoTFullAccess” and click Attach Policy:



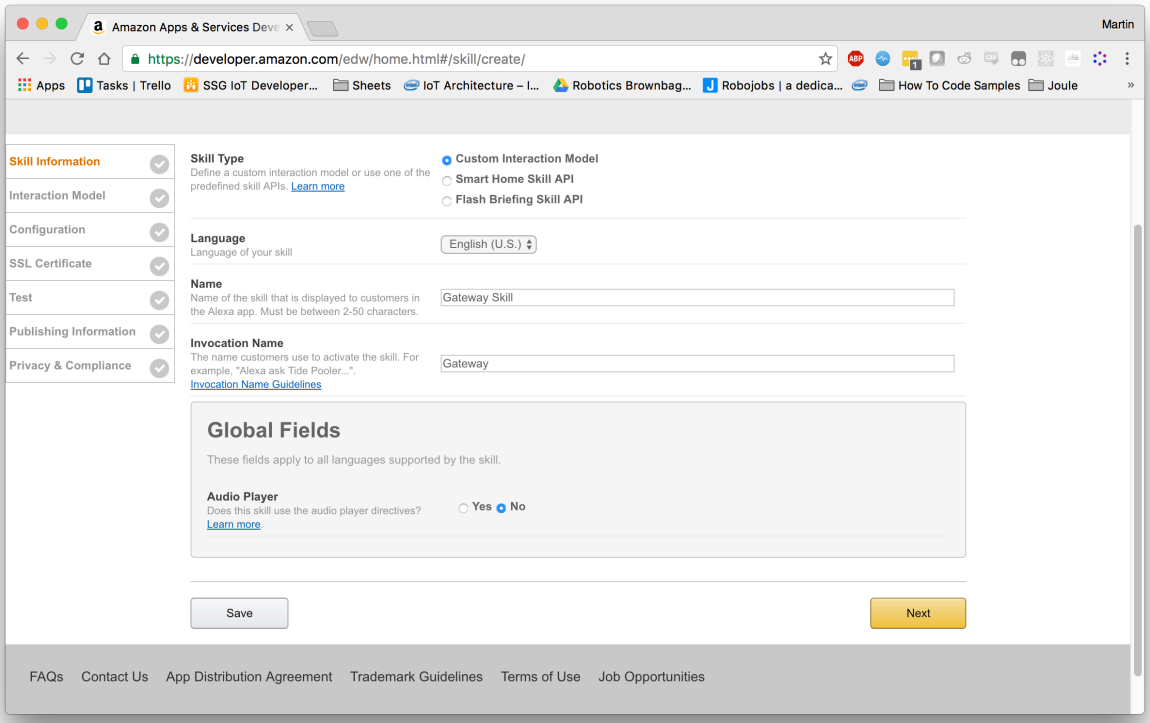
Configure Custom Alexa Skill

Finally, we will create a custom Alexa skill that will pass events to the handler running in the Lambda function.

Navigate to the Alexa developer portal: https://developer.amazon.com/edw/home.html#/ and select “Get Started” under Alexa Skills Kit



Select “Add a New Skill”. On the following screen add a name for your skill and an invocation name. The invocation name is how you will call the skill from Alexa e.g “Alexa tell Gateway to turn on blue LED” where “Gateway” is the invocation name.

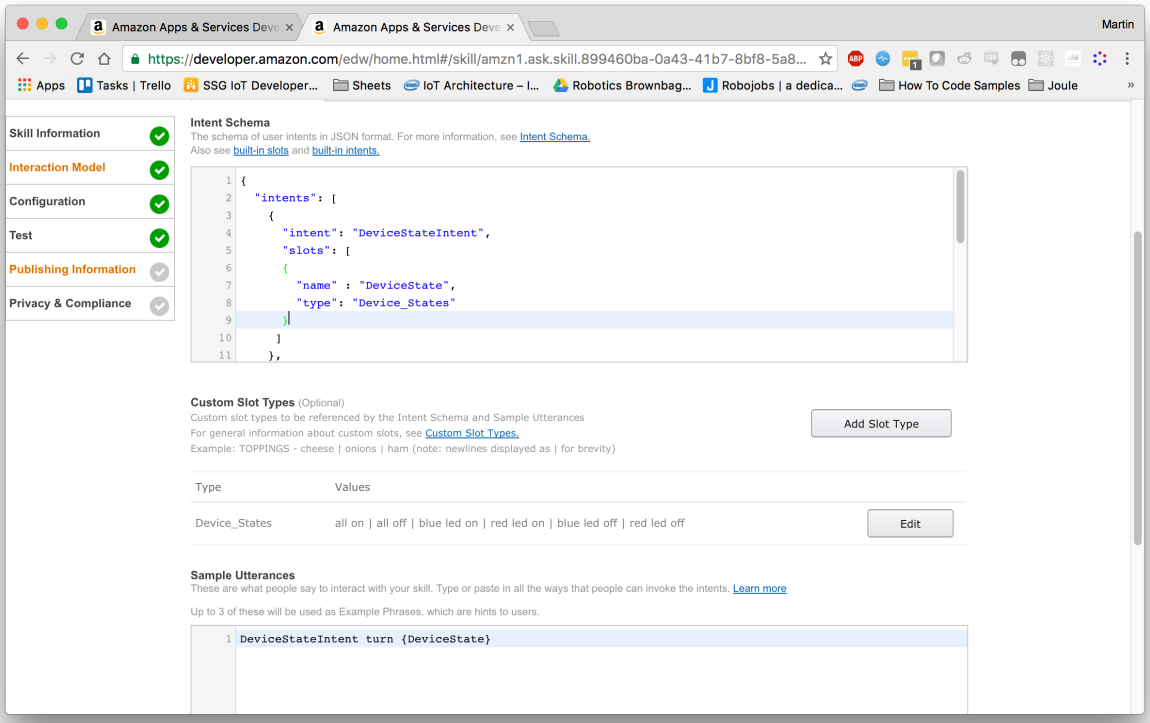


Click “Next”

On the next screen, we will configure the Intent Schema, add custom Slots, and set the utterance. The Intent Schema describes the events that are sent to Lambda and the slots associated with each. Slots can be thought of as variables for the device state, e.g. “blue led off” is a slot that describes the intended state of our gateway. The utterance is what people say to interact with your skill.

In Custom_Skill/Intent_Schema you will find the intent schema that we will use for this project. This schema describes an intent called “DeviceStateIntent” with Slots called “DeviceState” of Type “Device_States”. Note that DeviceStateIntent is the event name used in the handler lambda function for switching the state of the gateway – if you rename the event name in the handler function you must also rename it in the intent schema.

There are also some default helper intents in the schema that send events to the handler function when the user says things like “help”.



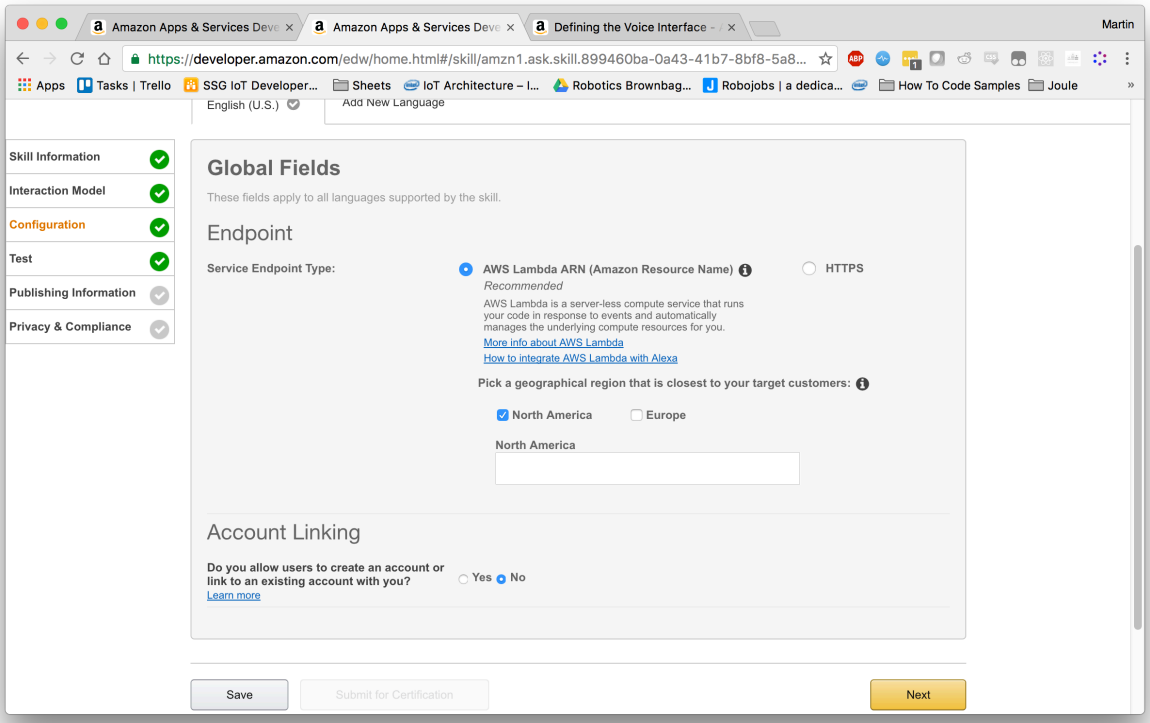
Device_States is not a standard slot type so we will have to define them. Select Add Slot Type, enter Device_States as the type and add the values separated by line, i.e. hit enter between each value. These values are the different cases passes to the DeviceStateIntent handler function. On line 56 of the lambda function you can see **case "all on":** which defines what the function does when “all on” is passed by the DeviceStateIntent event. We will use “all on”, “all off”, “blue led on”, “blue led off”, “red led on”, “red led off”. You may add your own, but you must define what slot does in the lambda handler function.

Finally add a Sample Utterance. The first term in the sample utterance defines which intent to tie the utterance to, i.e. the user does not say “DeviceStateIntent turn on blue led”, only “turn blue led on” and then “blue led on” is passed as the slot for DeviceStateIntent. Any terms in curly braces are slot names defined in the intent schma. Use the simplest form for now “Device StateIntent turn {DeviceState}” Note: Alexa is clever enough to substitute “turn” with “switch” and other contextual terms, no need to explicitly define it.

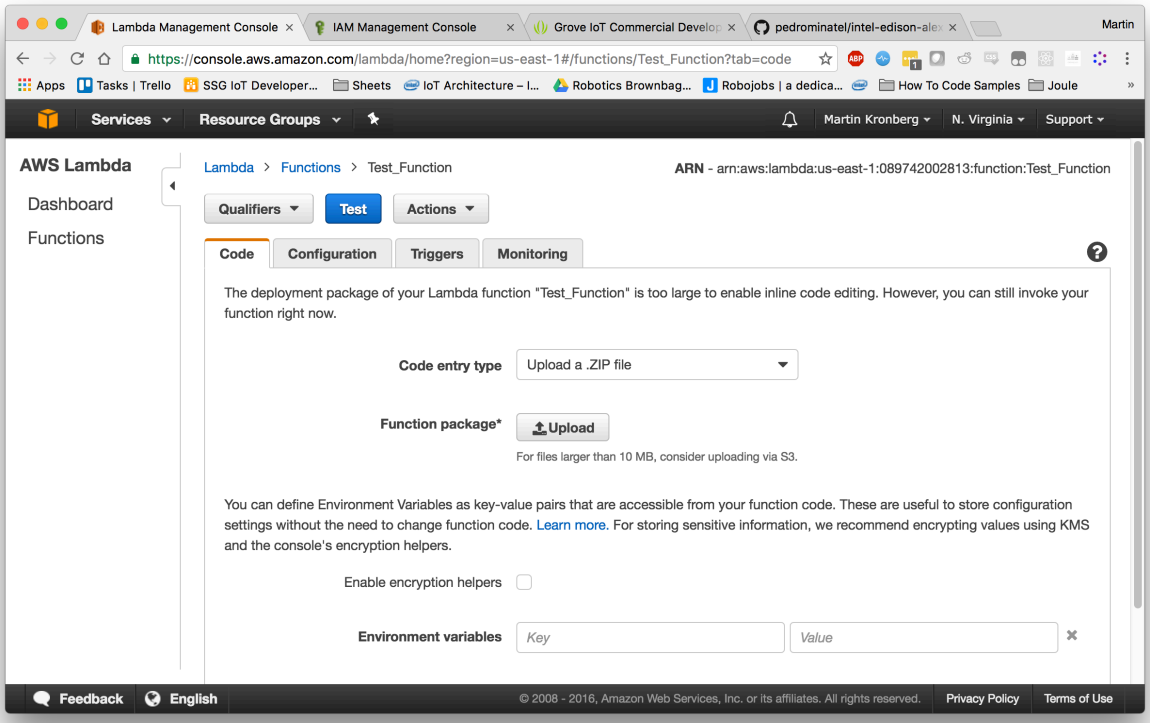
Much more info on intent schema setup can be found here:

https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/defining-the-voice-interface#The Sample Utterances File

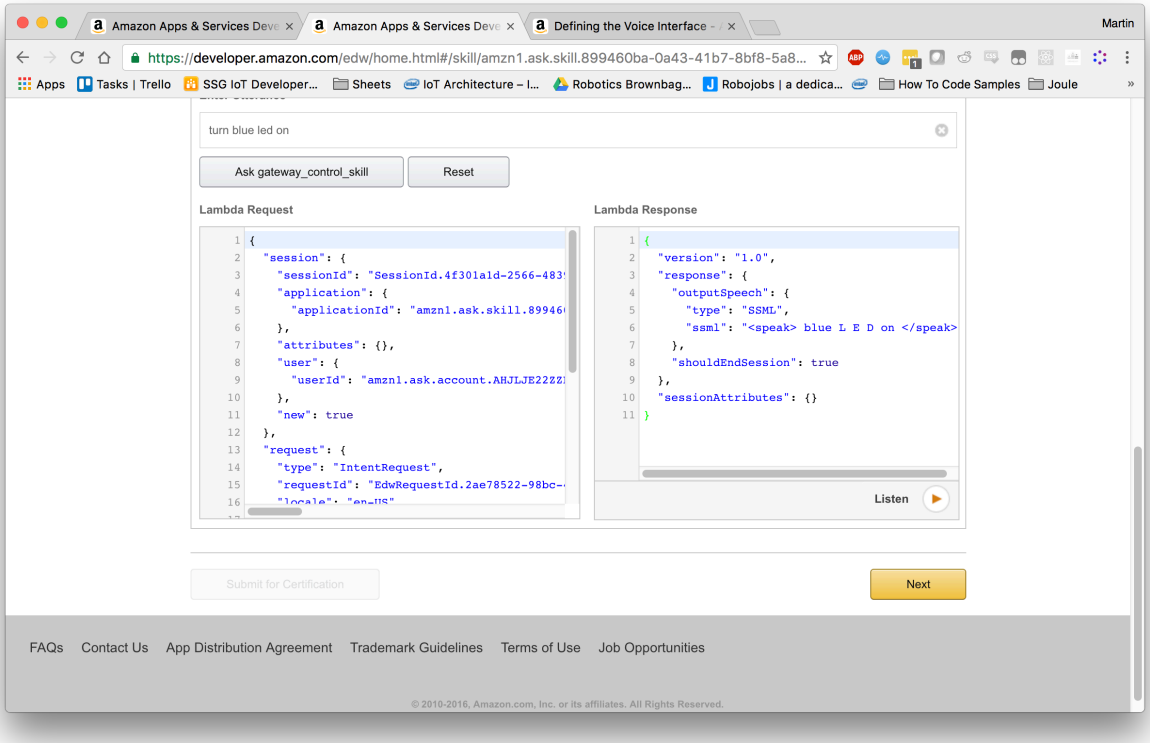
Click next to continue to the configuration.



Here select AWS Lambda ARN (Amazon Resource Name) as the endpoint type, and enter the ARN found in your Lambda Function Dashboard in the upper right:



Click next to go to test your new skill:



Enter an utterance like “turn blue led on” and click Ask. IF everything is working correctly you should see the lambda request and response with “Turn Blue LED on” as the output speech.

You are now ready to run the program:

You are now ready to run the program. Launch “Python main.py” from the root folder of the repository.

Wait 15 seconds for everything to initialize.

Click the button, say “Tell gateway to turn blue led on”

The Blue LED should turn on and Alexa will respond with “Blue LED on”