

Enable Amazon Alexa Voice Service (AVS)* on Intel® NUC

This tutorial explains steps to port Amazon Alexa Voice Service (AVS)* on Intel® NUC Gateway with Wind River Linux* operating system. This tutorial assumes you have already done some basic setup on your Intel® IoT Gateway and are familiar with its operation.

1. Setup Arduino 101* and Firmata:

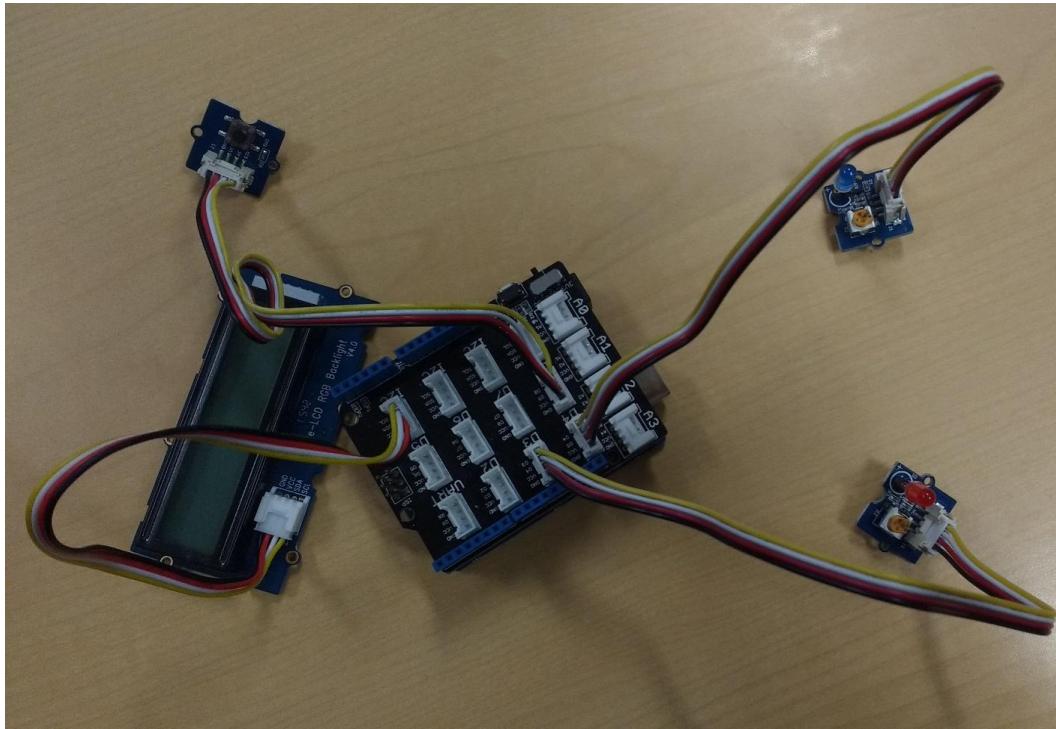
Clone the GitHub* repository onto your Intel® IoT Gateway by logging into the gateway and running:

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

2. Wire up the sensors as follows:

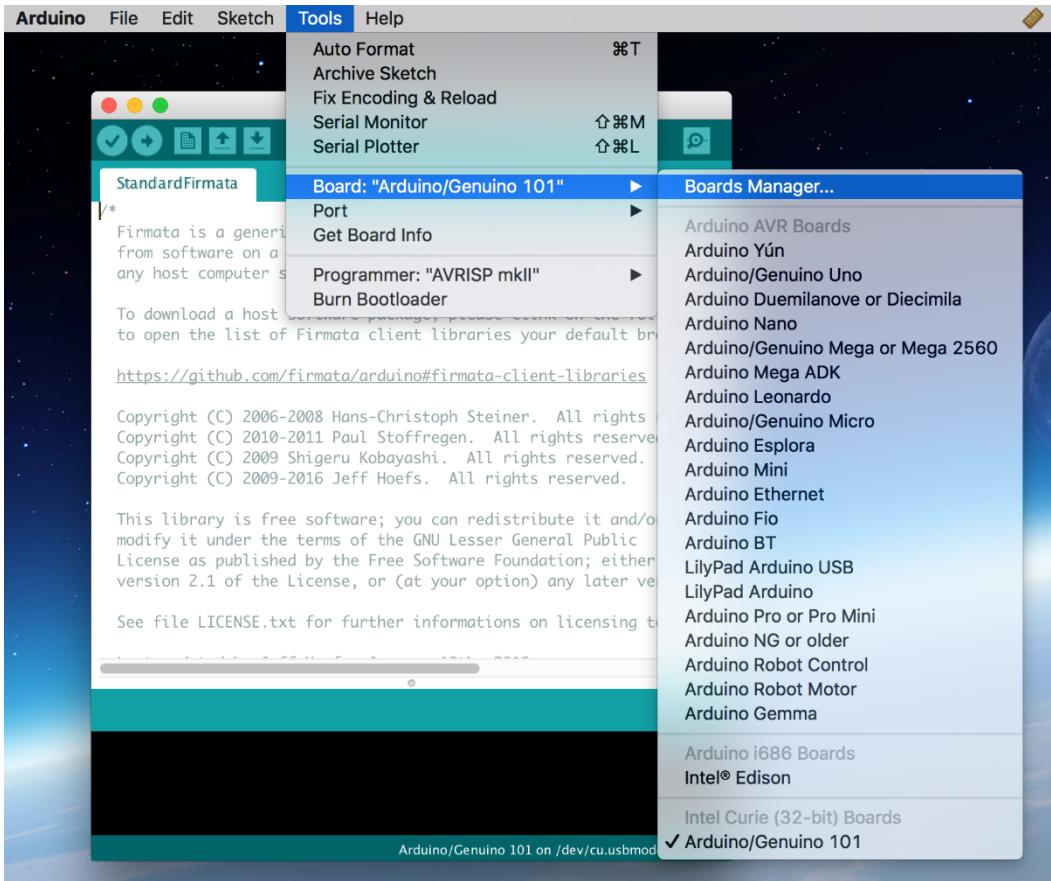
Connect Grove* button to pin D8, one LED to pin D3 and other one to Pin D4 as shown in the following picture using the cables from the Grove*

starter sensor kit box.

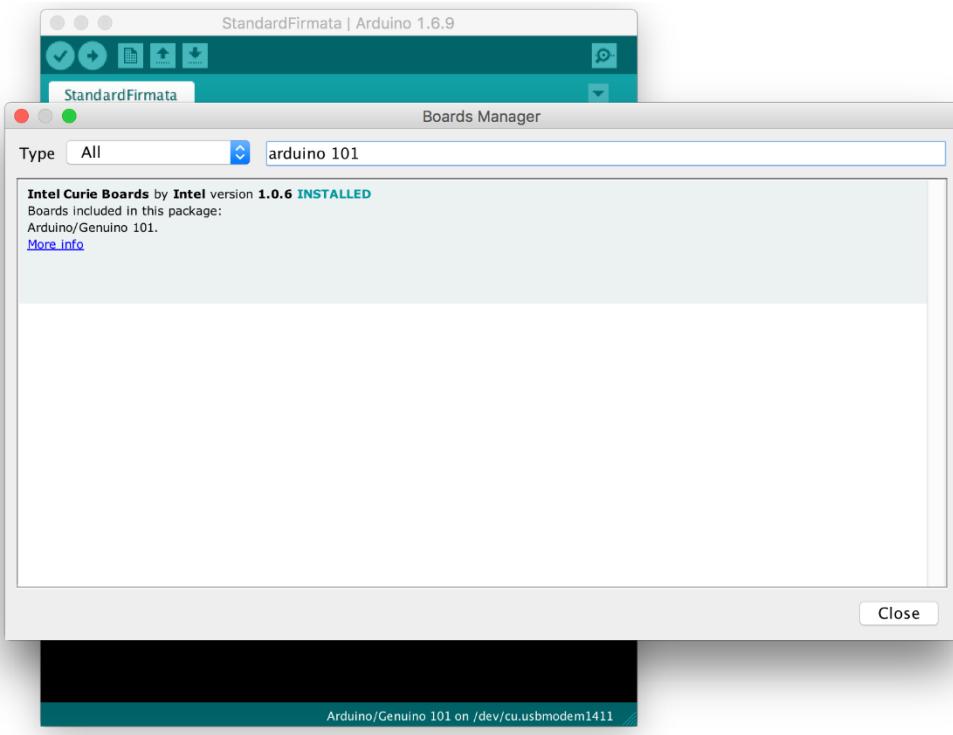


3. Set Up Firmata* on Arduino 101*:

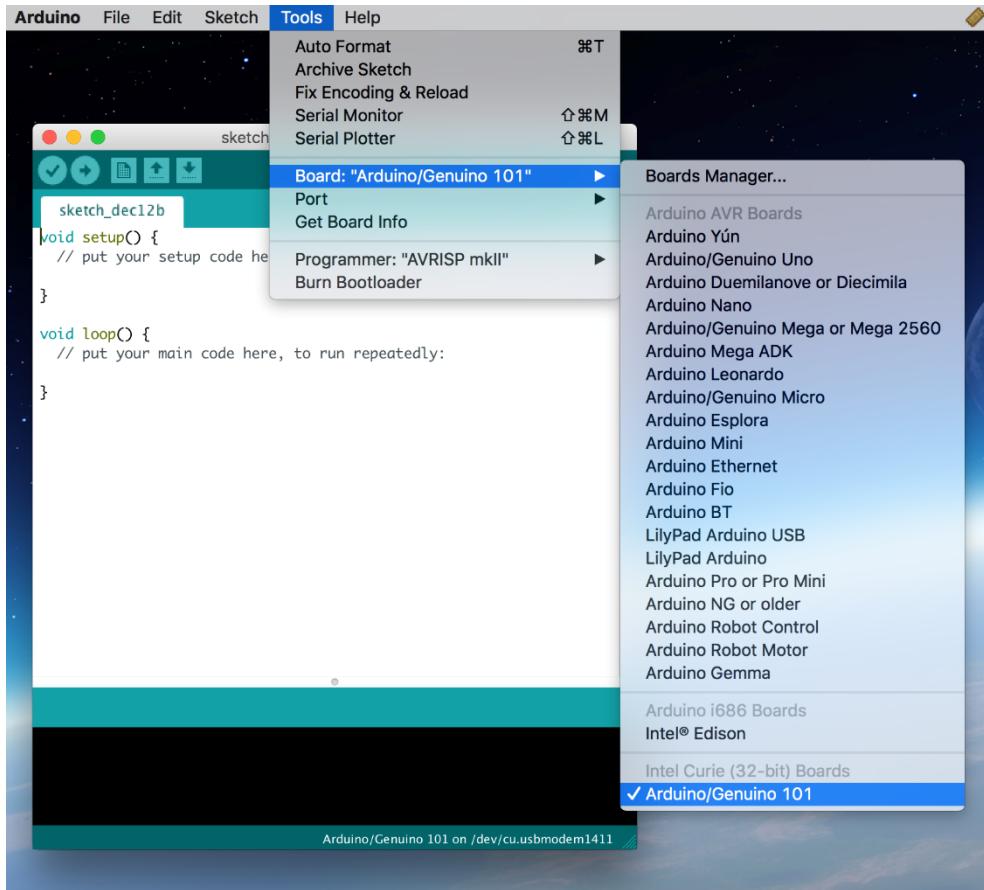
- a. Download Arduino* IDE: <https://www.arduino.cc/en/Main/Software>
- b. Connect Arduino 101* (branded Genuino 101* outside the U.S.) to a host computer via USB
- c. Launch Arduino IDE
- d. Open boards manager:



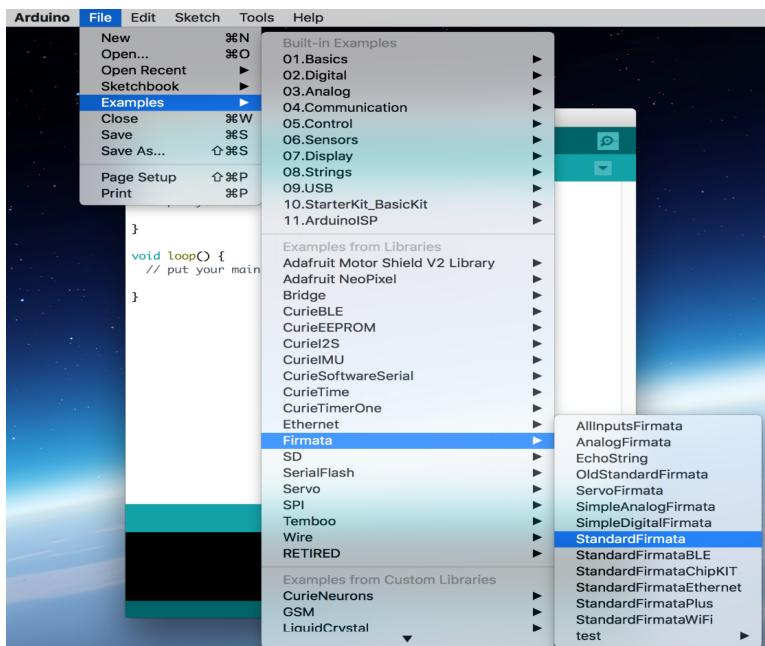
e. Search for “Arduino 101” and install the board definition:



- 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”

The screenshot shows the Arduino IDE interface with the title bar "StandardFirmata | Arduino 1.6.9". The main window displays the "StandardFirmata" sketch. The code includes a header section with a copyright notice for Firmata, followed by instructions to download client libraries from GitHub. It also contains standard GNU Lesser General Public License text and a reference to a LICENSE.txt file. At the bottom of the code area, there is a message "Done uploading." and a status bar indicating the sketch uses 34,300 bytes of program storage space. The status bar also shows the connection is "Arduino/Genuino 101 on /dev/cu.usbmodem1411".

```
/*
Firmata is a generic protocol for communicating with microcontrollers
from software on a host computer. It is intended to work with
any host computer software package.

To download a host software package, please click on the following
link 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 reserved.
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 153,216 bytes.
Starting download script...
SUCCESS: Sketch will execute in about 5 seconds.

Arduino/Genuino 101 on /dev/cu.usbmodem1411
```

4. Setup Gateway as Echo:

a. SSH onto the Gateway:

```
ssh root@IP.OF.GATE.WAY
```

Where IP.OF.GATE.WAY is the IP address of your gateway

Use “root” as the password

b. CD into the Repo:

```
cd ~/Alexa-on-Intel-NUC
```

c. Run setup.sh:

```
chmod +x ./setup.sh  
./setup.sh
```

The script will ask you to add a few channels, enter “y” and hit enter when prompted. The setup script will install the dependencies needed for the project and will take around 5 minutes to finish.

5. Reboot the Intel® IoT Gateway

6. Generate SSL Key for Intel® NUC

a. Run these commands:

```
openssl genrsa -out /root/privkey.pem 2048
```

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

Fill out your information for the SSL certificate.

7. Create an AWS Account:

Follow the instructions at this link to create an Amazon Web Services (AWS)* account and security profile: <https://github.com/alexa/alexa-avs-sample-app/wiki/Linux>

8. Security profile setup for Intel® NUC on Amazon* Developer console

- a. Login to Amazon Developer Portal - developer.amazon.com
- b. Go to APPS & SERVICES -> Security Profile
- c. Click Web setting tab

The screenshot shows the Amazon Developer Console interface. At the top, there's a navigation bar with links like DASHBOARD, APPS & SERVICES (which is highlighted), ALEXA, REPORTING, SUPPORT, DOCUMENTATION, and SETTINGS. Below the navigation bar, there's a secondary menu with links for My Apps, App Testing Service, Promotions, Security Profiles (which is also highlighted), Login with Amazon, Dash Replenishment Service, Alexa New, GameCircle, and PC / Mac & Web Instant Access. Under the 'Security Profiles' section, there's a sub-menu with Tester Management, Advertise Your App, and Mobile Ads. The main content area is titled 'Security Profile Management' and shows a specific 'Alexa Voice Service Sample App Security Profile - Security Profile'. This profile has tabs for General, Web Settings (which is selected), Android/Kindle Settings, and iOS Settings. The 'Web Settings' tab shows fields for Allowed Origins (set to https://localhost:3000) and Allowed Return URLs (containing https://localhost:3000/authresponse and https://192.168.1.197:5000/code). A red box labeled '2' is placed over the second URL in the list. A red box labeled '1' is placed over the 'Edit' button in the bottom right corner of the 'Web Settings' form.

- d. Click Edit button on Web settings tab.
- e. Get IP address of your Intel® NUC using “ifconfig” command.
- f. Add the URL in the form https://ip_address_for_NUC:5000/code to Allowed Return URLs
e.g. <https://192.168.1.197:5000/code>

9. Add Your Credentials to Creds.py

- a. Navigate to the General tab on the Security Profile Management page:

The screenshot shows the Amazon Developer Console interface. At the top, there's a navigation bar with links for DASHBOARD, APPS & SERVICES (which is highlighted), ALEXA, REPORTING, SUPPORT, DOCUMENTATION, and SETTINGS. On the right side of the top bar, it shows the user's name 'PRIYANKA - INTEL' and options to SIGN OUT and change the ENGLISH language. Below the top bar, there's a secondary navigation menu with links for My Apps, App Testing Service, Promotions, Security Profiles (which is also highlighted in orange), Login with Amazon, Dash Replenishment Service, Alexa New, GameCircle, and PC / Mac & Web Instant Access. At the bottom of this menu, there are links for Tester Management, Advertise Your App, and Mobile Ads.

Security Profile Management

[More Information](#)
[Login with Amazon](#)
[GameCircle](#)
[Device Messaging](#)

Alexa Voice Service Sample App Security Profile - Security Profile

[General](#) [Web Settings](#) [Android/Kindle Settings](#) [iOS Settings](#)

These settings apply to all the apps using this security profile. Your security profile credentials — Client ID and Client Secret — allow your app to securely identify itself to Amazon services. [Learn More](#)

Security Profile Name Alexa Voice Service Sample App Security Profile

Security Profile Description Alexa Voice Service Sample App Security Profile Description

Security Profile ID [REDACTED]

Client ID [REDACTED]

Client Secret [REDACTED]

Consent Privacy Notice URL <http://example.com>

Consent Logo Image

- b. Open creds.py using vi with:
vi creds.py
- c. Hit "a" to enter insert mode
- d. Enter your information into ProductID, Client_ID, etc.
- e. Hit "esc", type ":wq", and hit enter to save the file

Creating Custom Skills For The Gateway to control sensors:

10. 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.

- a. Create an AWS account: <https://console.aws.amazon.com>

b. From Services select AWS IoT:

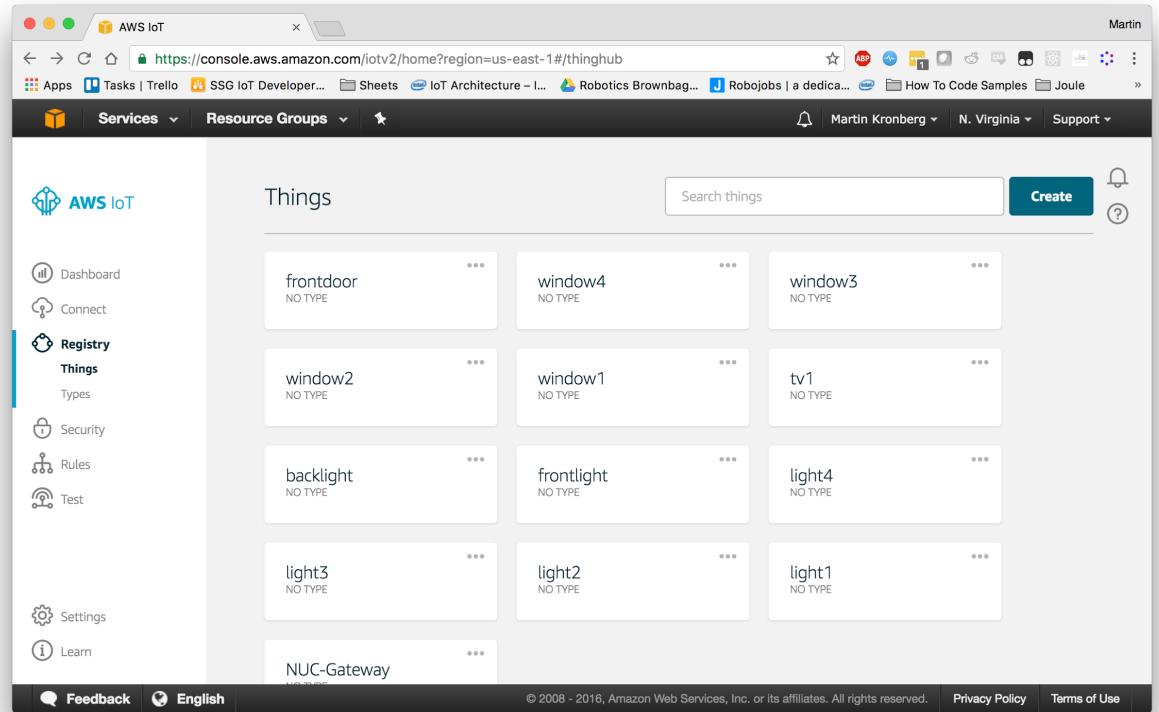
Make sure that your region is selected as N. Virginia (us-east-1). The AWS services are not yet online for all regions.

The screenshot shows the AWS Management Console Services dashboard. At the top right, the region is set to "N. Virginia". The "Internet Of Things" service is highlighted with a red box. Other services listed include Compute, Storage, Database, Networking & Content Delivery, Migration, Developer Tools, Management Tools, Analytics, Artificial Intelligence, Security, Identity & Compliance, Game Development, Mobile Services, Application Services, Messaging, Business Productivity, Desktop & App Streaming, and others.

The dashboard should look like this:

The screenshot shows the AWS IoT Dashboard. On the left, there is a sidebar with links: Connect, Registry, Security, Rules, Test, Settings, and Learn. The main area displays a chart titled "Successful connections" showing data from December 15 to December 21. The chart shows a peak on December 20th with approximately 40 successful connections, followed by a dip on December 21st. Below the chart, there is a section titled "Messages".

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



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

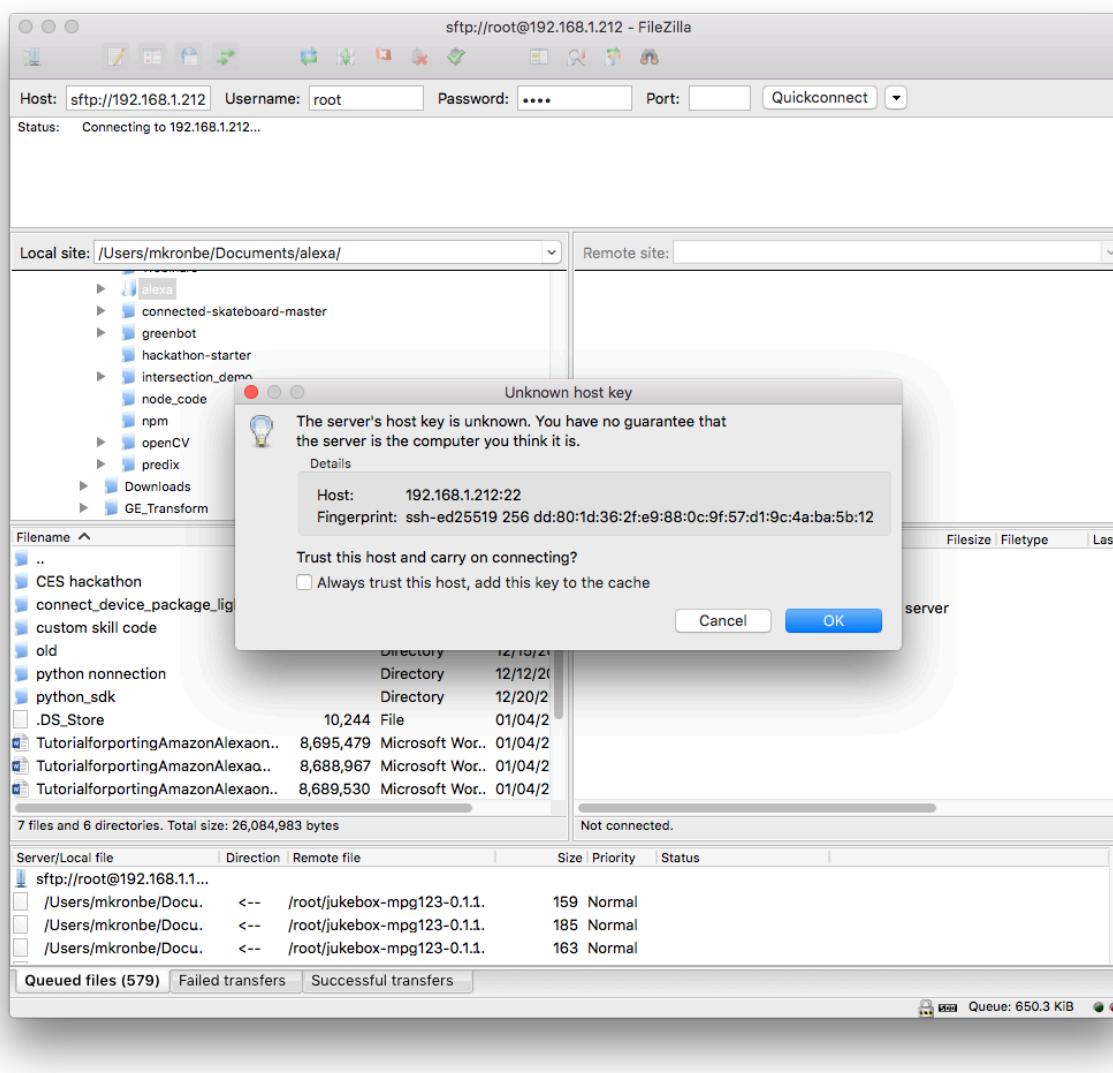
The screenshot shows the AWS IoT Thing details page for a thing named 'Gateway2'. The top navigation bar includes links for 'AWS IoT', 'Services', 'Resource Groups', and user information ('Martin Kronberg', 'N. Virginia', 'Support'). The main content area is titled 'THING' and shows 'Gateway2' with 'NO TYPE'. A sidebar on the left lists 'Details', 'Security', 'Shadow', 'Interact' (which is selected), and 'Activity'. The 'Interact' section contains a 'Connect a device' button (highlighted with an orange border) and an 'HTTPS' section with a 'Rest API Endpoint' URL: `a21a7zf3kffmrf.iot.us-east-1.amazonaws.com`. Below that is an 'MQTT' section with a 'Rest API Endpoint' URL: `$aws/things/Gateway2/shadow/update`.

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

The screenshot shows the 'Choose a platform' step in a tutorial for porting Amazon services. The top navigation bar is identical to the previous screenshot. The main content area is titled 'Choose a platform' and shows two options: 'Linux/OSX' (highlighted with a blue background) and 'Windows'. Below this, under 'Choose a AWS IoT Device SDK', it says 'Connect in 15 minutes or less with these SDKs and a quick setup script.' It lists three options: 'Node.js', 'Python', and 'Java', each with a right-pointing arrow indicating further steps.

- f. 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.
- g. Transfer the zip file to the gateway using FTP from the command line, or FileZilla:

- Go to <https://filezilla-project.org/> to download and install the FileZilla client
- Open Filezilla and connect to the IP of your Gateway using “root” as username, “root” as password, and 22 as the port:

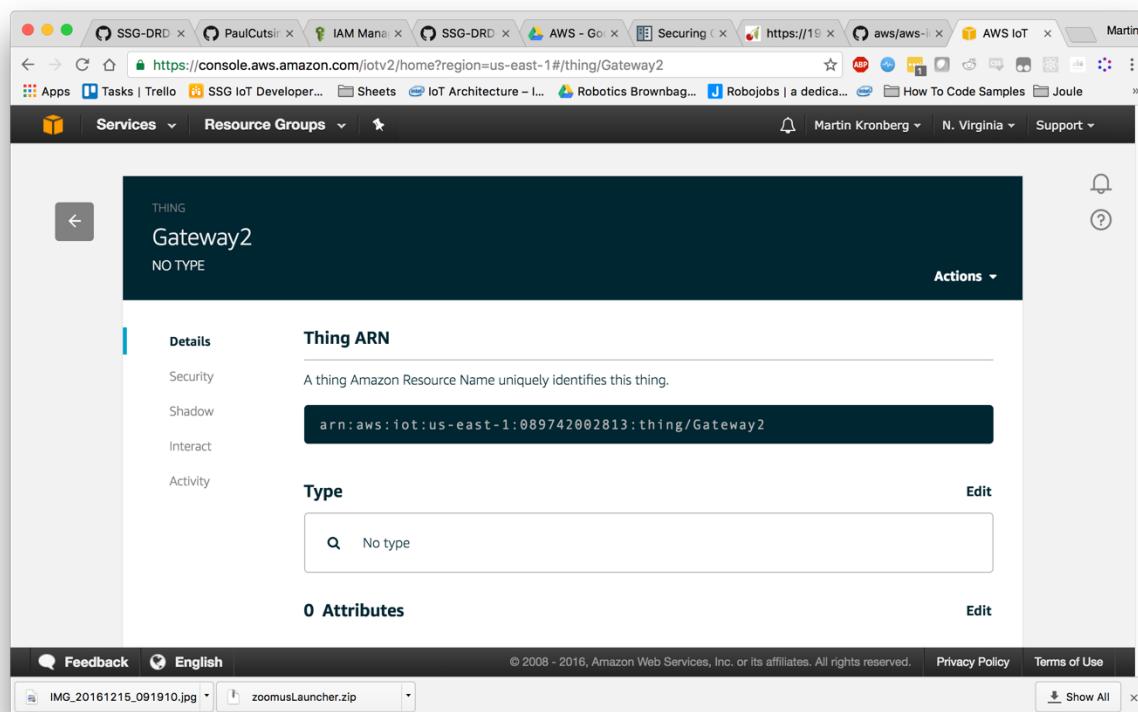


- When asked about unknown host key hit OK
- Navigate to the zip file on the left panel and the Alexa-on-Intel-NUC repository folder on the right panel. Double click the zip file to transfer it

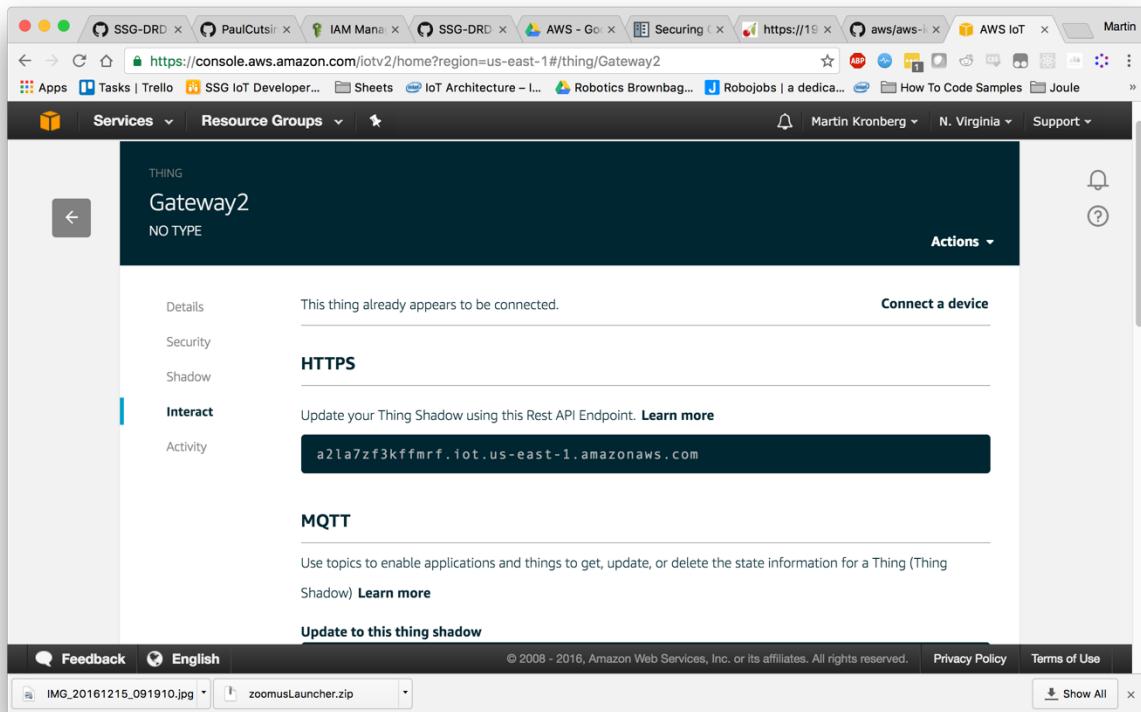
11. Edit main.py:

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.

- a. Using VI open main.py and replace AWS_IOT_ARN with the Thing arn from the Details section of the AWS IoT device console:



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



12. 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.

- a. 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**

The screenshot shows the AWS IoT Thing Details page for a device named 'NUC-Gateway'. The left sidebar has tabs for 'Details', 'Security', 'Shadow', and 'Interact' (which is selected). The main content area shows the Thing Shadow endpoint as 'iot.us-east-1.amazonaws.com' highlighted with a yellow box. Other sections include 'HTTPS' and 'MQTT'.

- b. 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.
- c. Navigate to the AWS Lambda service: <https://console.aws.amazon.com/lambda/>
- d. Click 'Create Function' use the Blank Function blueprint:

Lambda Management Console

https://console.aws.amazon.com/lambda/home?region=us-east-1#/create/select-blueprint

Services Resource Groups

Martin Kronberg N. Virginia Support

Lambda > New function

Select blueprint

Configure triggers

Configure function

Review

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your desired scenario and customize as needed, or skip this step if you want to author a Lambda function and configure an event source separately. Except where otherwise noted, blueprints are licensed under CC0.

Select runtime Filter Viewing 1-9 of 70

Blank Function	kinesis-firehose-syslog-to-json	alexa-skill-kit-sdk-factskill
Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard. custom	An Amazon Kinesis Firehose stream processor that converts input records from RFC3164 Syslog format to JSON. nodejs - kinesis-firehose	Demonstrate a basic fact skill built with the ASK NodeJS SDK nodejs - alexa
kinesis-firehose-apachelog-to...	cloudfront-modify-response-h...	s3-get-object-python

e. Select “Alexa Skills Kit” as the trigger:

Lambda Management Console

https://console.aws.amazon.com/lambda/home?region=us-east-1#/create/configure-triggers

Services Resource Groups

Martin Kronberg N. Virginia Support

Lambda > New function

Select blueprint

Configure triggers

Configure function

Review

Configure triggers

You can choose to add a trigger that will invoke your function.

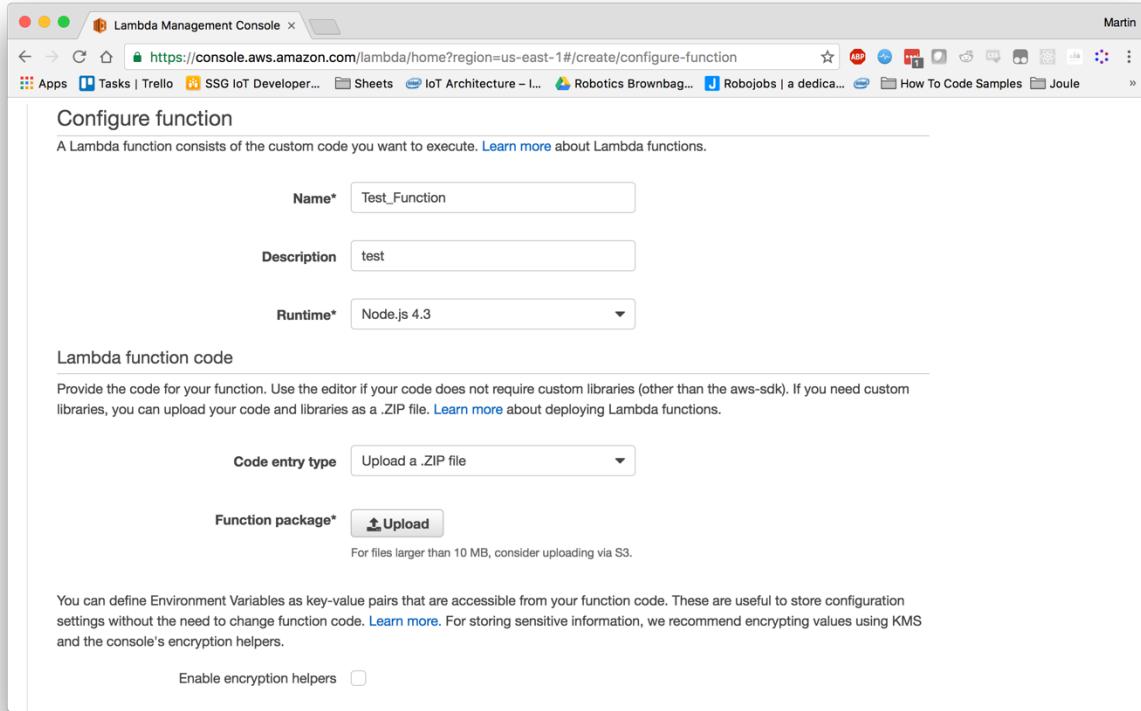
Alexa Skills Kit → Lambda Remove

Choosing Submit will create a resource policy that allows the Amazon Alexa service to call your Lambda function. To configure the Alexa service to work with your Lambda function, go to the [Alexa Developer portal](#). [Learn more](#) about the Lambda permission model.

Cancel Previous Next

Feedback English © 2008 – 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

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



- g. Click Upload and select the zip file you made earlier.

- h. Enter index.handler for the Handler – this is the default handler for Lambda
i. Choose “Create new role from template”
j. Select “Simple Microservice Permissions” from “Policy Templates”

The screenshot shows the AWS Lambda Management Console interface for creating a new function. The 'Lambda function handler and role' section is active. Key configuration parameters shown include:

- Handler**: index.handler
- Role***: Create new role from template(s)
- Role name***: Lambda_IoT_Role
- Policy templates**: Simple Microservice per...

In the 'Advanced settings' section, the following values are set:

- Memory (MB)***: 128
- Timeout***: 0 min 3 sec

A note at the bottom of this section states: "AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's [retry policy](#) and [DLQs](#). Please ensure your role has appropriate permissions to access the DLQ resource."

- k. Leave everything else as default and click next.
- l. Review the function on the next page and click “Create Function”

13. Configure IAM access policy:

- a. Navigate to AWS IAM console: <https://console.aws.amazon.com/iam>
- b. Navigate to the Roles list form the left side menu:

The screenshot shows the AWS IAM Management Console interface. The left sidebar has a 'Roles' section selected. The main area displays a table of roles with columns for 'Role Name' and 'Creation Time'. One role, 'aws_iot_sns', is highlighted with a blue selection bar. The top navigation bar shows tabs for Lambda Management Console, IAM Management Console, and another Lambda Management Console tab. The URL in the address bar is https://console.aws.amazon.com/iam/home?region=us-east-1#/roles.

Role Name	Creation Time
aws_iot_dynamoDB	2016-06-08 11:02 PST
aws_iot_sns	2016-05-26 13:12 PST
dynamoDB_role	2016-06-09 10:09 PST
IAM_Debug	2016-12-09 14:31 PST
LambdaIoT	2016-12-14 16:09 PST
lambda_basic_execution	2016-06-09 15:00 PST
Lambda_IoT_Role	2016-12-21 13:53 PST

- c. Select the role policy you created for your Lambda function
- d. Select “Attach Policy”

The screenshot shows the AWS IAM Management Console interface. On the left, there's a sidebar with navigation links: Dashboard, Groups, Users, Roles (which is selected), Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area has a title 'Summary' and displays the following details for the role:

- Role ARN:** arn:aws:iam::089742002813:role/service-role/Lambda_IoT_Role
- Instance Profile ARN(s):** None
- Path:** /service-role/
- Creation Time:** 2016-12-21 13:53 PST

Below this, there are tabs for **Permissions**, **Trust Relationships**, **Access Advisor**, and **Revoke Sessions**. The **Permissions** tab is active, showing a section titled 'Managed Policies' with the following content:

The following managed policies are attached to this role. You can attach up to 10 managed policies.

Attach Policy

Policy Name	Actions
AWSLambdaBasicExecutionRole-12e8af73-170e-4bba-928e-9c8fb311c655	Show Policy Detach Policy Simulate Policy
AWSLambdaMicroserviceExecutionRole-17c67a9c-bb2b-4a48-9f9a-cb197e2990b9	Show Policy Detach Policy Simulate Policy

At the bottom of the page, there are links for Feedback, English, and a copyright notice: © 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. There are also links for Privacy Policy and Terms of Use.

e. Select “AWSIoTFullAccess” and click Attach Policy:

The screenshot shows the 'Attach Policy' dialog box. On the left, it says 'Attach Policy'. The main area is titled 'Attach Policy' and contains the following text: 'Select one or more policies to attach. Each role can have up to 10 policies attached.' Below this is a search bar with 'Filter: Policy Type' set to 'AWS' and a results count of 'Showing 98 results'. A table lists policies with columns: Policy Name, Attached Entities, Creation Time, and Edited Time. The policy 'AWSIoTFullAccess' is selected (indicated by a checked checkbox). The table rows are:

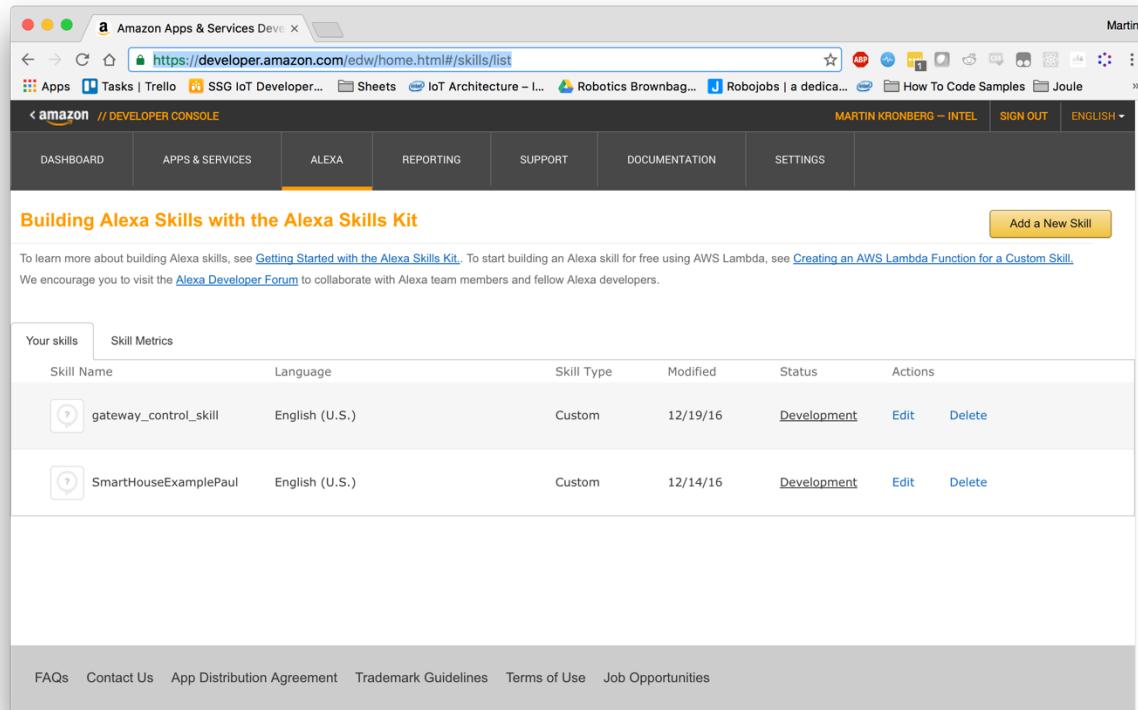
	Policy Name	Attached Entities	Creation Time	Edited Time
<input type="checkbox"/>	aws-iot-role-logging_-18...	1	2016-12-09 14:31 PST	2016-12-09 14:31 PST
<input checked="" type="checkbox"/>	AWSIoTFullAccess	1	2015-10-08 08:19 PST	2015-10-08 08:19 PST
<input type="checkbox"/>	AWSLambdaBasicExecu...	1	2016-12-14 16:09 PST	2016-12-14 16:09 PST
<input type="checkbox"/>	AWSLambdaMicroservic...	1	2016-12-14 16:09 PST	2016-12-14 16:09 PST
<input type="checkbox"/>	AmazonEC2RoleforAWS...	0	2015-05-19 11:10 PST	2015-05-19 11:10 PST
<input type="checkbox"/>	AWSAccountActivityAcc...	0	2015-02-06 10:41 PST	2015-02-06 10:41 PST
<input type="checkbox"/>	AWSAccountUsageRepo...	0	2015-02-06 10:41 PST	2015-02-06 10:41 PST
<input type="checkbox"/>	AWSAgentlessDiscovery...	0	2016-08-01 18:35 PST	2016-08-01 18:35 PST
<input type="checkbox"/>	AWSApplicationDiscover...	0	2016-05-11 14:38 PST	2016-05-11 14:38 PST

At the bottom right of the dialog are 'Cancel' and 'Attach Policy' buttons.

14. Configure Custom Alexa Skill

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

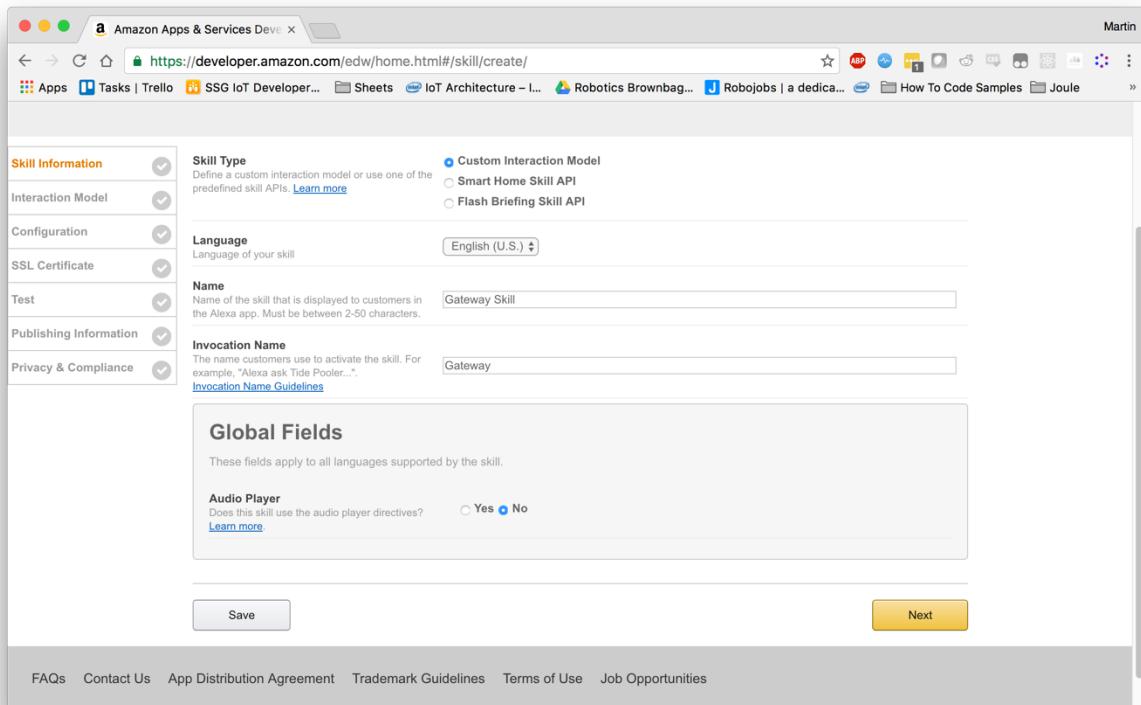
- a. Navigate to the Alexa developer portal: <https://developer.amazon.com/edw/home.html#/skills/list> and select “Get Started” under Alexa Skills Kit



The screenshot shows a web browser window for the Alexa Skills Kit developer console. The URL in the address bar is <https://developer.amazon.com/edw/home.html#/skills/list>. The page title is "Building Alexa Skills with the Alexa Skills Kit". At the top right, there is a sign-out link for "MARTIN KRONBERG – INTEL" and a language selection dropdown set to "ENGLISH". Below the title, there is a yellow button labeled "Add a New Skill". A note below the button says: "To learn more about building Alexa skills, see [Getting Started with the Alexa Skills Kit](#). To start building an Alexa skill for free using AWS Lambda, see [Creating an AWS Lambda Function for a Custom Skill](#). We encourage you to visit the [Alexa Developer Forum](#) to collaborate with Alexa team members and fellow Alexa developers." The main content area displays a table titled "Your skills" with two rows of data. The columns are "Skill Name", "Language", "Skill Type", "Modified", "Status", and "Actions". The first row has a skill name of "gateway_control_skill", a language of "English (U.S.)", a skill type of "Custom", a modified date of "12/19/16", a status of "Development", and actions "Edit" and "Delete". The second row has a skill name of "SmartHouseExamplePaul", a language of "English (U.S.)", a skill type of "Custom", a modified date of "12/14/16", a status of "Development", and actions "Edit" and "Delete". At the bottom of the page, there are links for "FAQs", "Contact Us", "App Distribution Agreement", "Trademark Guidelines", "Terms of Use", and "Job Opportunities".

Your skills	Skill Metrics					
Skill Name	Language	Skill Type	Modified	Status	Actions	
gateway_control_skill	English (U.S.)	Custom	12/19/16	Development	Edit	Delete
SmartHouseExamplePaul	English (U.S.)	Custom	12/14/16	Development	Edit	Delete

- b. 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.



- c. Click “Next”
- d. 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.
- e. 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”.

The screenshot shows the Intent Schema configuration page for an Alexa skill. The Intent Schema section displays the following JSON code:

```

1 {
2   "intents": [
3     {
4       "intent": "DeviceStateIntent",
5       "slots": [
6         {
7           "name": "DeviceState",
8           "type": "Device_States"
9         }
10      ]
11    }
12  }

```

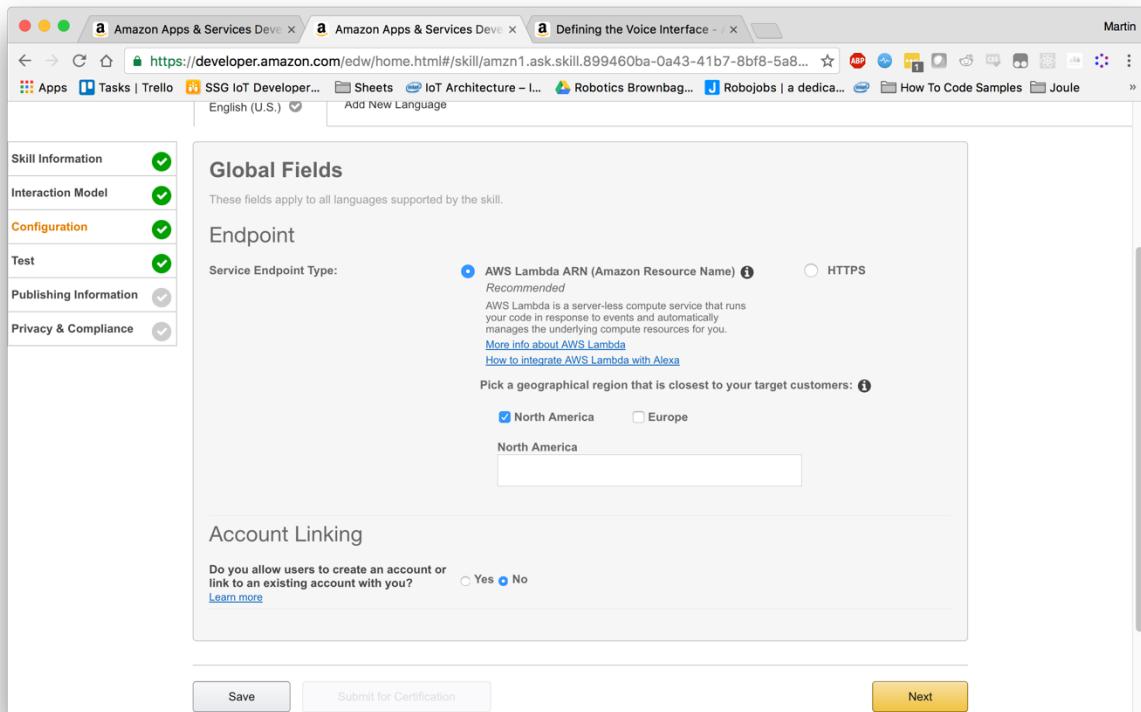
The Custom Slot Types section shows a table with one entry:

Type	Values
Device_States	all on all off blue led on red led on blue led off red led off

The Sample Utterances section contains the following example:

```
1 DeviceStateIntent turn {DeviceState}
```

- f. 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.
- g. 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 schema. 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.
- h. 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>
- i. Click next to continue to the configuration.



- j. 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:

The deployment package of your Lambda function "Test_Function" is too large to enable inline code editing. However, you can still invoke your function right now.

Code entry type: Upload a .ZIP file

Function package*:

For files larger than 10 MB, consider uploading via S3.

You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#). For storing sensitive information, we recommend encrypting values using KMS and the console's encryption helpers.

Enable encryption helpers

Environment variables

k. Click next to go to test your new skill:

turn blue led

Ask gateway_control_skill Reset

Lambda Request

```

1 {
2   "session": {
3     "sessionId": "SessionId.4f301ald-2566-483",
4     "application": {
5       "applicationId": "amzn1.ask.skill.899460ba-0a43-41b7-8bf8-5a8..."
6     },
7     "attributes": {},
8     "user": {
9       "userId": "amzn1.ask.account.AHJLJE2ZZZ"
10    },
11    "new": true
12  },
13  "request": {
14    "type": "IntentRequest",
15    "requestId": "EdwRequestId.2ae78522-98bc-"
16    "locale": "en-US"
17 }
  
```

Lambda Response

```

1 {
2   "version": "1.0",
3   "response": {
4     "outputSpeech": {
5       "type": "SSML",
6       "ssml": "<speak> blue L E D on </speak>"
7     },
8     "shouldEndSession": true
9   },
10   "sessionAttributes": {}
11 }
  
```

Listen

Submit for Certification

FAQs Contact Us App Distribution Agreement Trademark Guidelines Terms of Use Job Opportunities

© 2010-2016, Amazon.com, Inc. or its affiliates. All Rights Reserved.

- I. 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.

15. Run the program:

- a. You are now ready to run the program. Launch “Python main.py” from the root folder of the repository.
- b. Wait 15 seconds for everything to initialize.
- c. Click the button, say “Tell gateway to turn blue led on”
- d. The Blue LED should turn on and Alexa will respond with “Blue LED on”