

# 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:**

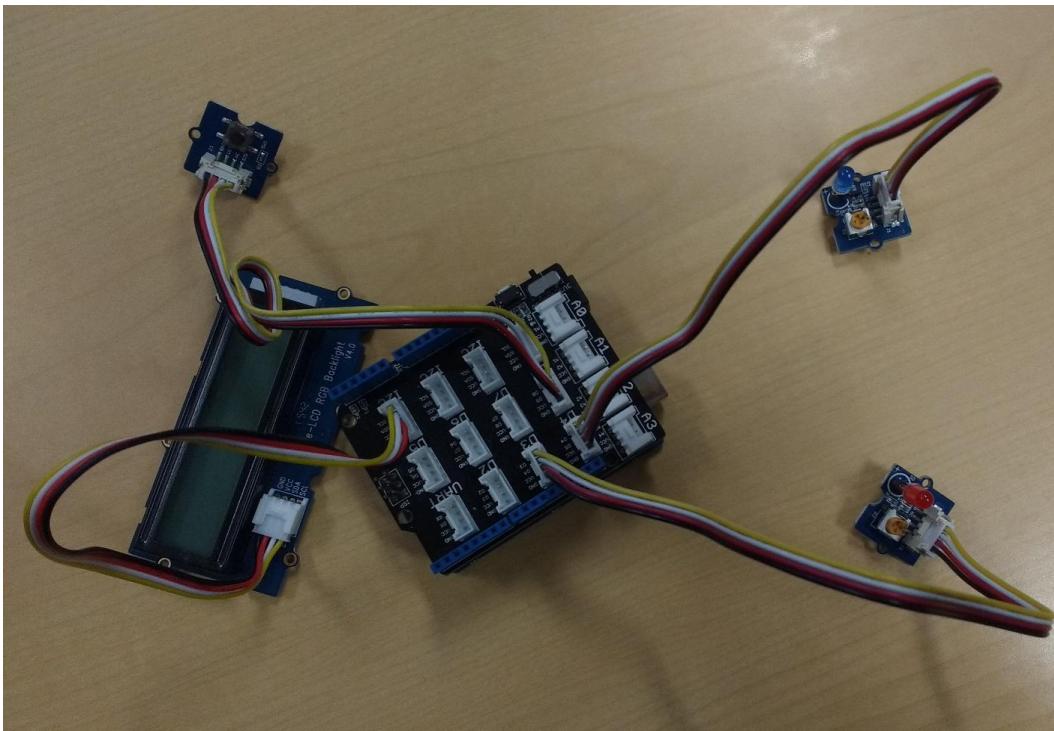
Clone the Github repository onto your gateway by logging into the gateway and running:

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

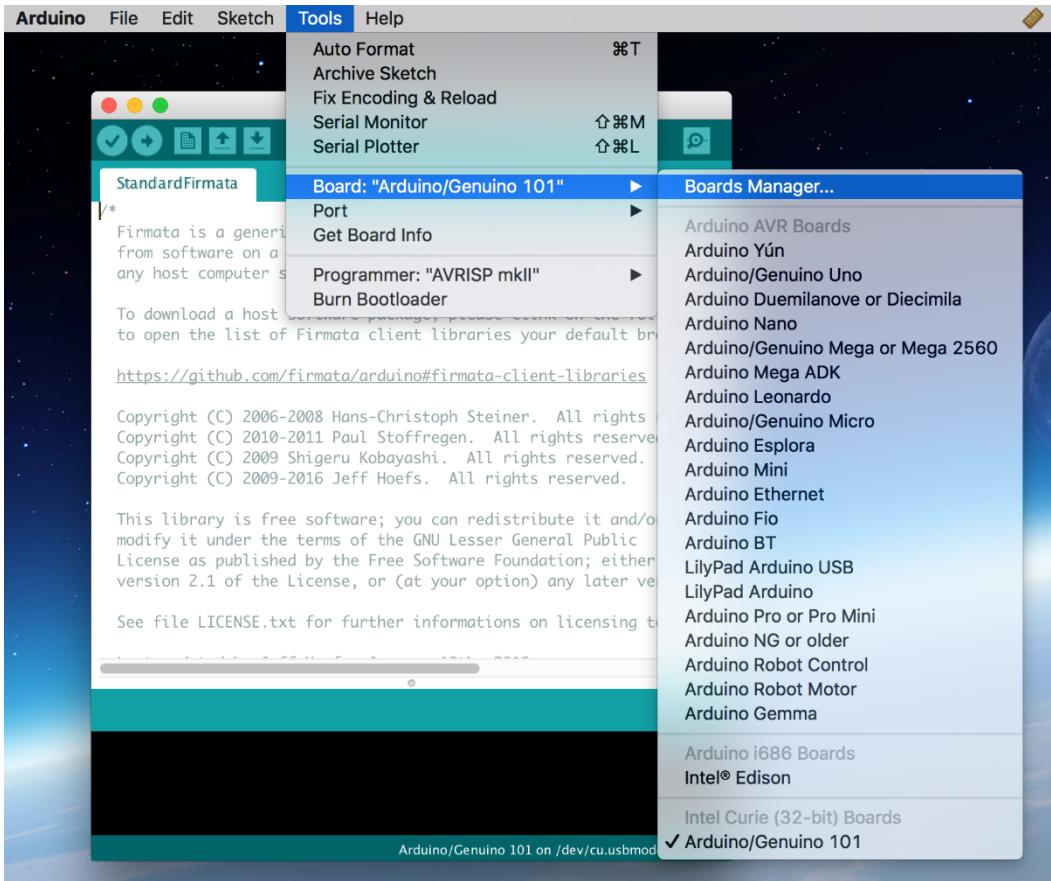
### **Wire up the sensors as follows:**

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

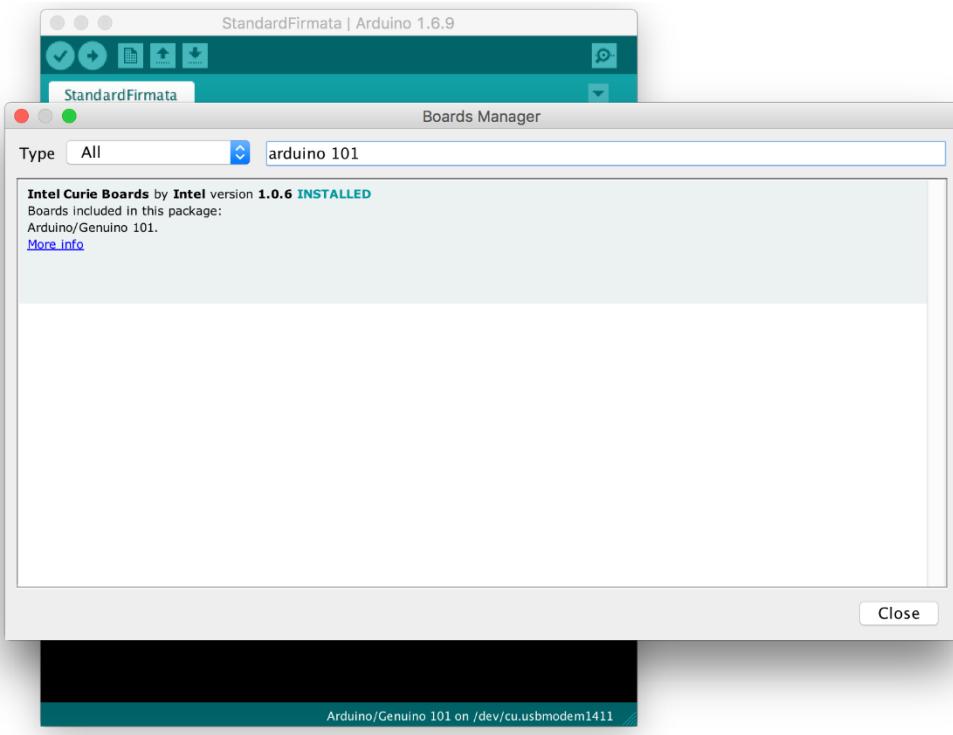
sensor kit box.



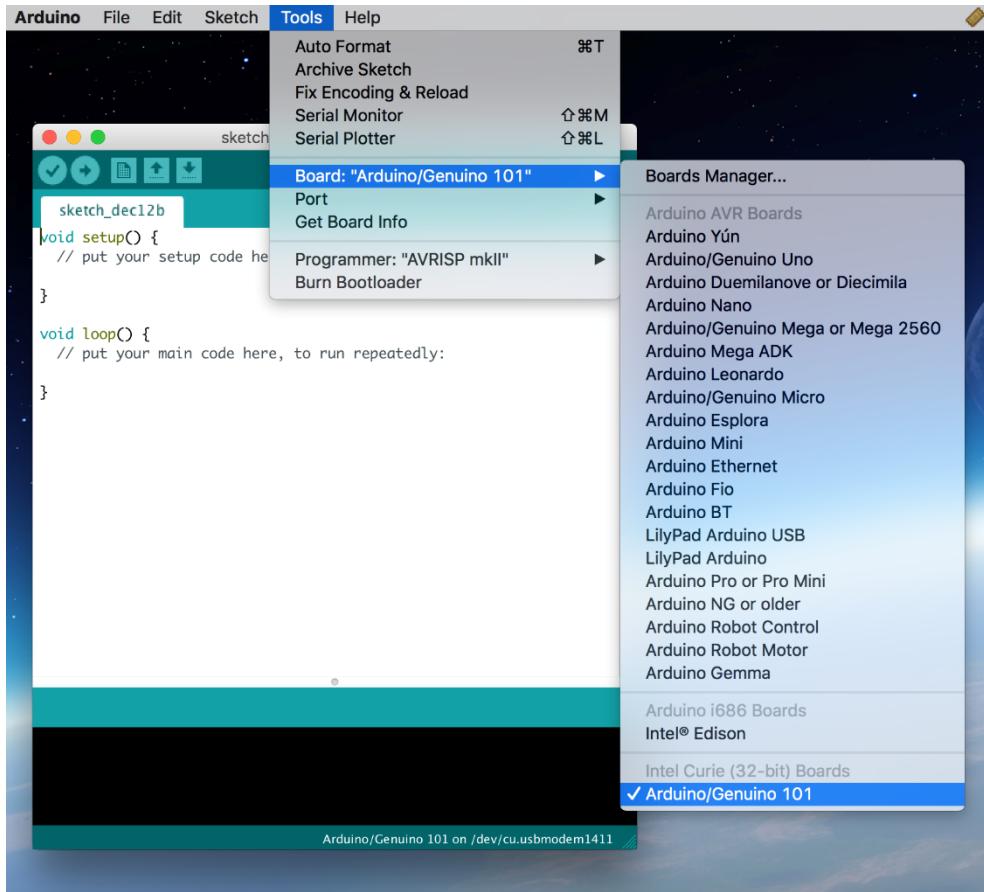
- Set Up Firmata on Arduino 101:
  - a. Download Arduino IDE: <https://www.arduino.cc/en/Main/Software>
  - b. Connect Arduino 101 to host computer via USB
  - c. Launch Arduino IDE
  - d. Open boards manager:



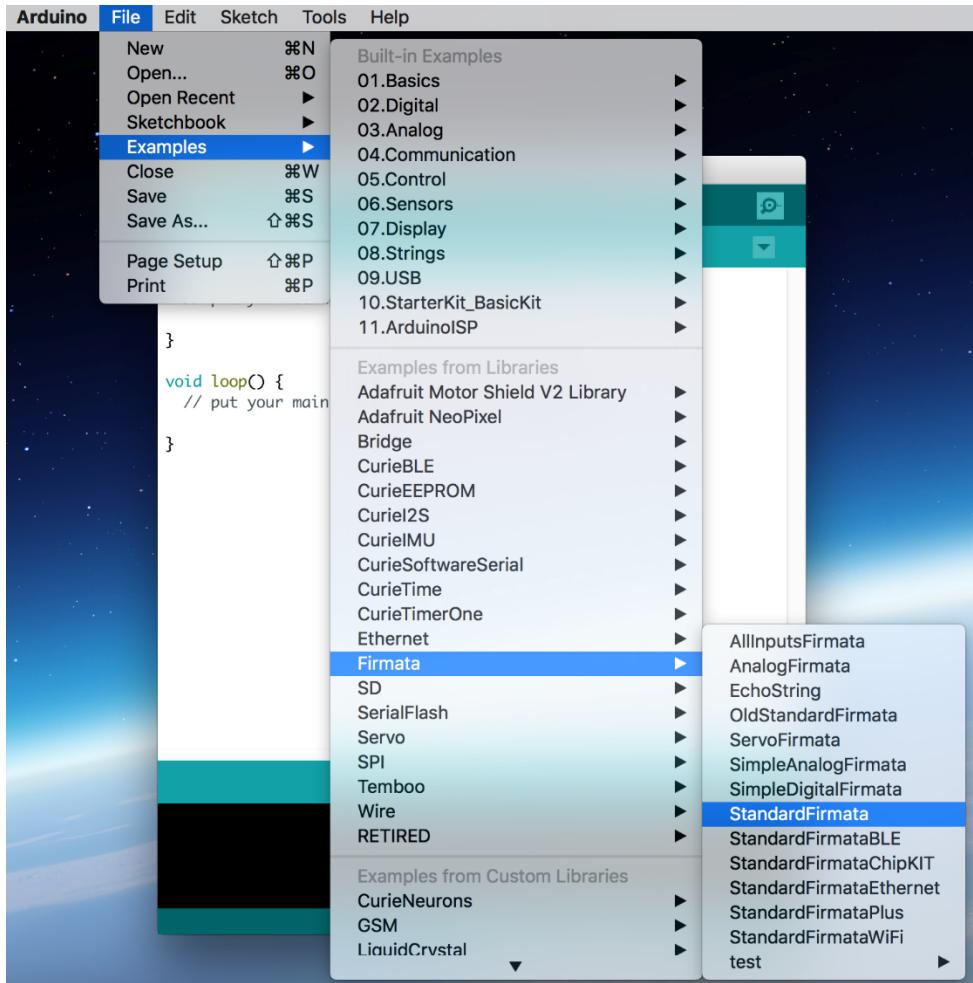
e. Search for “Arduino 101” and install the board definition:



- 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 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 158,960 bytes.
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

PRIYANKA – INTEL | SIGN OUT | ENGLISH ▾

DASHBOARD APPS & SERVICES ALEXA REPORTING SUPPORT DOCUMENTATION SETTINGS

My Apps App Testing Service Promotions Security Profiles Login with Amazon Dash Replenishment Service Alexa New GameCircle PC / Mac & Web Instant Access

Tester Management Advertise Your App 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

To use Login with Amazon with a website, you must specify either an allowed JavaScript origin (for the Implicit grant) or an allowed return URL (for the Authorization Code grant). If you are using Pay with Amazon, you must specify an allowed JavaScript origin. [Learn More](#)

Allowed Origins ? <https://localhost:3000>

Allowed Return URLs ? <https://localhost:3000/authresponse>  
2 <https://192.168.1.197:5000/code>

1 Edit

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](https://ip_address_for_NUC:5000/code)

e.g. <https://192.168.1.197:5000/code>

PRIYANKA – INTEL | SIGN OUT | ENGLISH ▾

DASHBOARD APPS & SERVICES ALEXA REPORTING SUPPORT DOCUMENTATION SETTINGS

My Apps App Testing Service Promotions Security Profiles Login with Amazon Dash Replenishment Service Alexa New GameCircle PC / Mac & Web Instant Access

Tester Management Advertise Your App 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 ?

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:

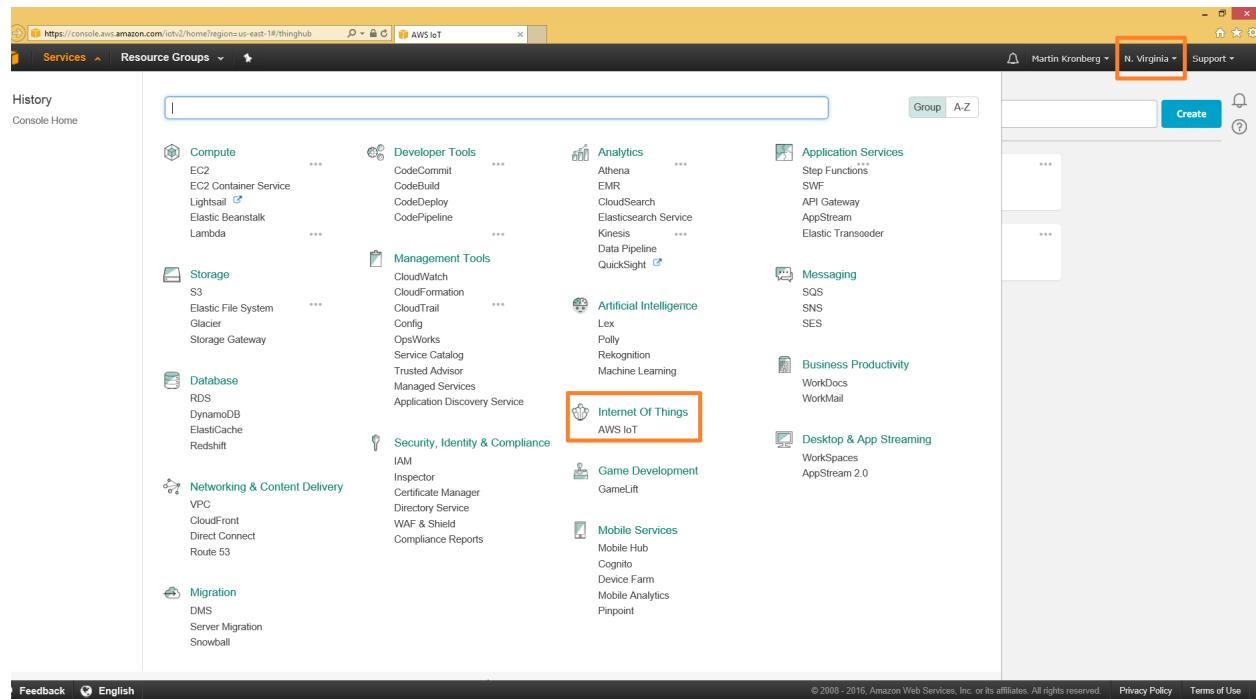
## Create an AWS IoT Device:

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.

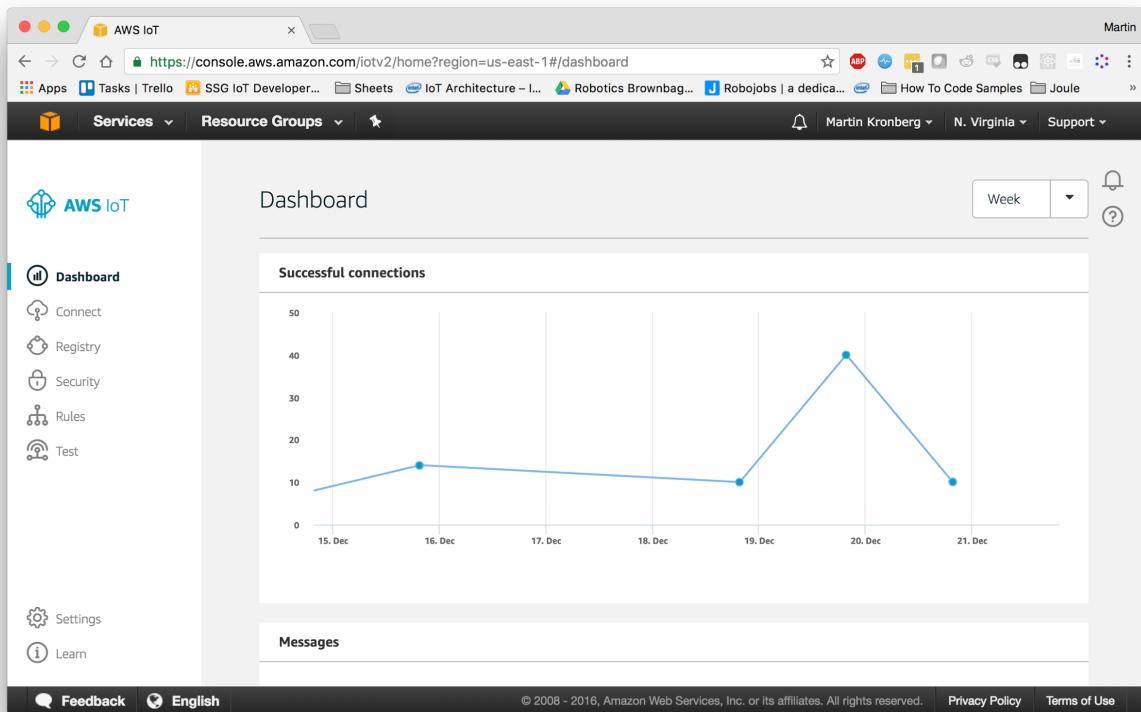
Create an AWS account: <https://console.aws.amazon.com>

From Services select AWS IoT:

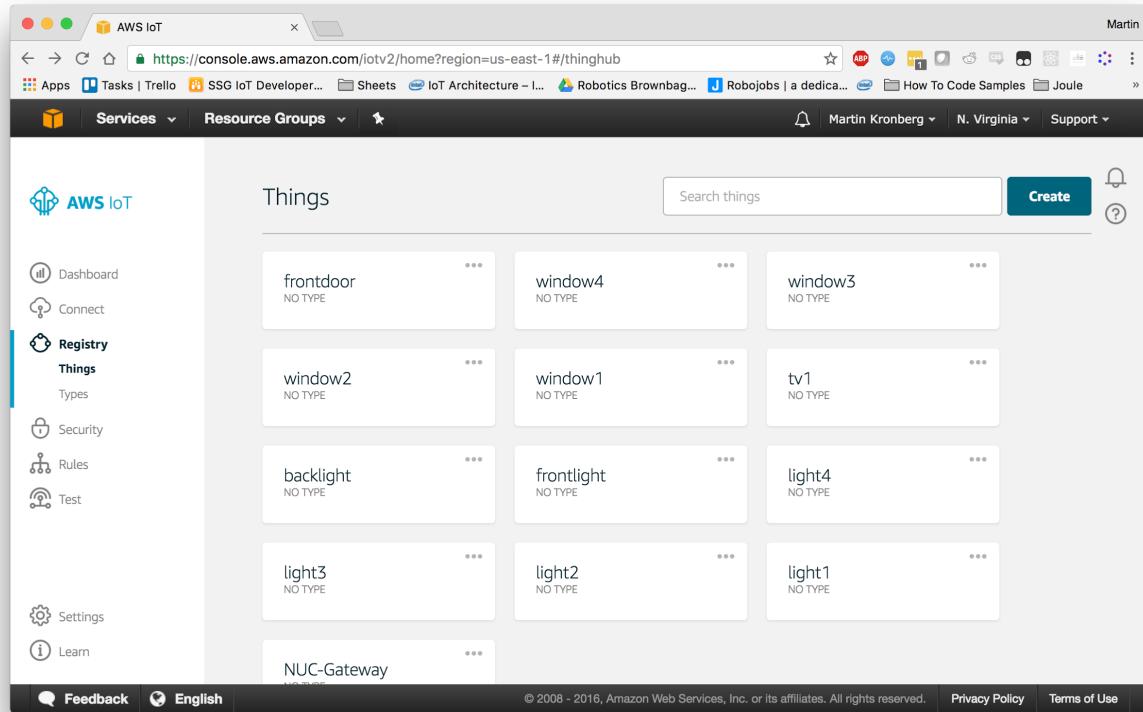
Make sure that your region is selected as N. Virginia (us-east-1). The AWS services are not yet online for all regions.



The dashboard should look like this:



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



The screenshot shows the AWS IoT Things page. On the left, there's a sidebar with icons for Dashboard, Connect, Registry (which is selected), Types, Security, Rules, Test, Settings, and Learn. The main area is titled "Things" and contains a search bar and a "Create" button. Below the search bar is a table of device entries:

Device Name	Type
frontdoor	NO TYPE
window4	NO TYPE
window3	NO TYPE
window2	NO TYPE
window1	NO TYPE
tv1	NO TYPE
backlight	NO TYPE
frontlight	NO TYPE
light4	NO TYPE
light3	NO TYPE
light2	NO TYPE
light1	NO TYPE
NUC-Gateway	NO TYPE

At the bottom of the page, there are links for Feedback, English, Privacy Policy, and Terms of Use.

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

The screenshot shows the AWS IoT Thing details page for a device named 'Gateway2'. The top navigation bar includes links for 'Services', 'Resource Groups', and 'Actions'. The main content area is titled 'THING' and shows 'Gateway2' with 'NO TYPE'. A message states 'This thing already appears to be connected.' with a 'Connect a device' button. Below this, the 'Interact' section contains 'HTTPS' and 'MQTT' tabs. The 'HTTPS' tab has a sub-section for updating the Thing Shadow via a REST API endpoint, showing the URL 'a21a7zf3kffmrf.iot.us-east-1.amazonaws.com'. The 'MQTT' tab provides instructions for enabling topics to get, update, or delete state information, with a 'Update to this thing shadow' command '\$aws/things/Gateway2/shadow/update'. The bottom of the page features standard AWS footer links for 'Feedback', 'English', 'Privacy Policy', and 'Terms of Use'.

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

The screenshot shows the 'Choose a platform' step of a tutorial. The top navigation bar is identical to the previous screenshot. The main content area is titled 'Choose a platform' and displays two options: 'Linux/OSX' and 'Windows'. Below this, a section titled 'Choose a AWS IoT Device SDK' encourages users to connect in 15 minutes using Node.js, Python, or Java. Each SDK option has a corresponding 'Next' button. The bottom of the page includes the usual AWS footer links.

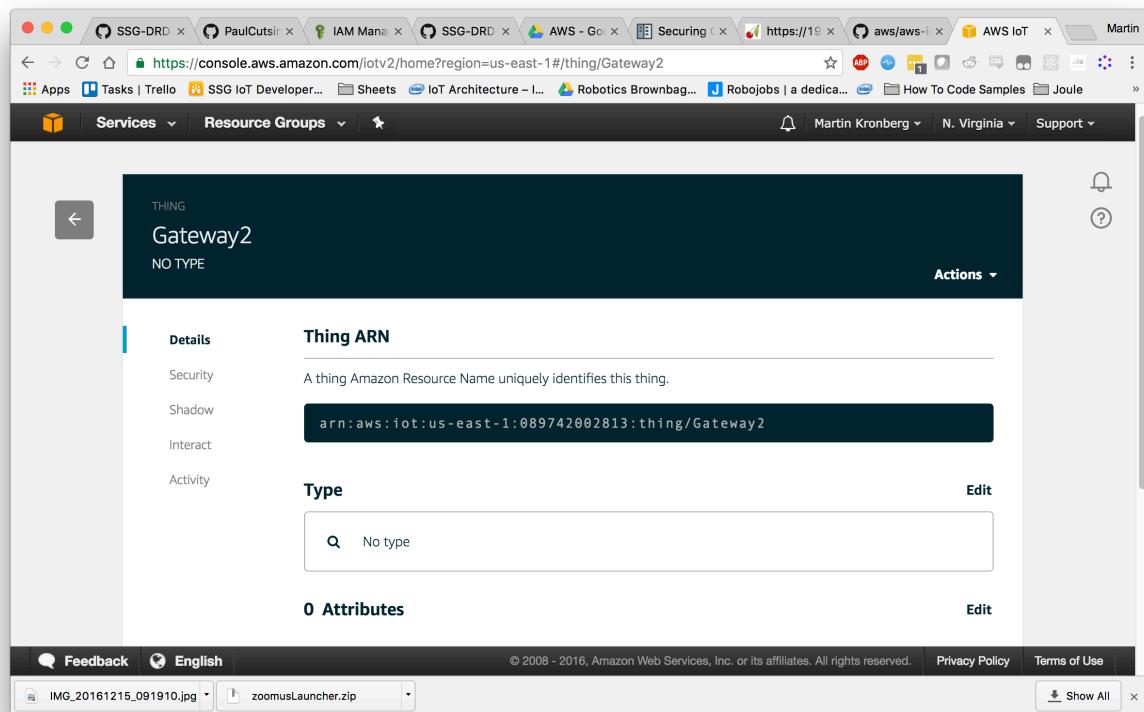
Click “Getting Started” and download the connection kit for your selected OS. This package will contain a private key and certificate which we will use to connect the gateway to the AWS IoT service. It also contains a start.sh script which will run a basic connection test, install the SDK, and download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

The screenshot shows the AWS IoT Device Management console. A tab in the browser bar says "https://console.aws.amazon.com/iotv2/home?region=us-east-1#/thing/Gateway2". The main page displays a "THING" card for "Gateway2" with "NO TYPE". The "Interact" tab is selected. Under "HTTPS", it says "Update your Thing Shadow using this Rest API Endpoint" and provides the endpoint "a21a7zf3kffmr.r.iot.us-east-1.amazonaws.com". Under "MQTT", it says "Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)" and provides a link to "Learn more". At the bottom, there are download links for "IMG\_20161215\_091910.jpg" and "zoomusLauncher.zip".

## 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**

The screenshot shows the AWS IoT Thing details page for 'NUC-Gateway'. The left sidebar has tabs for Details, Security, Shadow, Interact (which is selected), and Activity. 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.

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:

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

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.

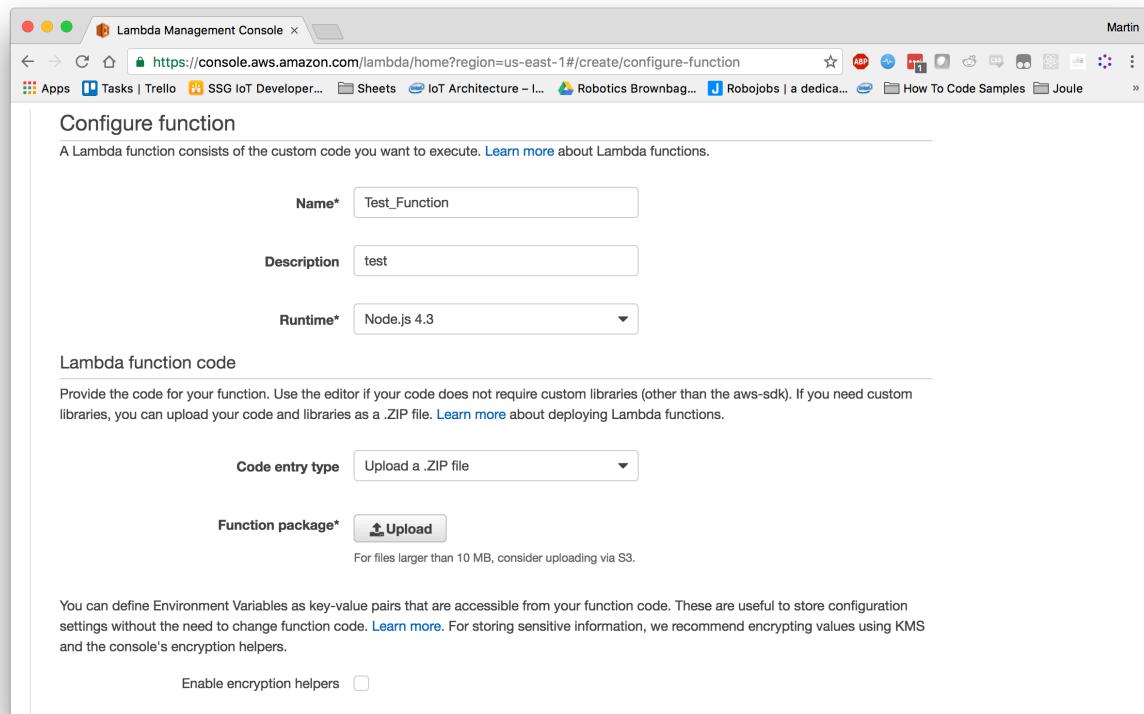
Alexa Skills Kit → Lambda Remove

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

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 from template”

Select “Simple Microservice Permissions” from “Policy Templates”

The screenshot shows the AWS Lambda Management Console interface. The URL in the address bar is <https://console.aws.amazon.com/lambda/home?region=us-east-1#/create/configure-function>. The page is titled "Lambda function handler and role". It contains several input fields:

- Handler\***: index.handler
- Role\***: Create new role from template(s) (dropdown menu)
- Role name\***: Lambda\_IoT\_Role
- Policy templates**: Simple Microservice per...

Below these fields is a section titled "Advanced settings" with the following configuration:

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

A note at the bottom of the advanced settings 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."

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:

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. The table shows the following data:

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

Select the role policy you created for your Lambda function

Select “Attach Policy”

The screenshot shows the AWS IAM Management Console. On the left, a sidebar menu is open under the 'Roles' section, listing 'Policies', 'Identity providers', 'Account settings', 'Credential report', and 'Encryption keys'. The main content area displays the 'Summary' tab for the role 'Lambda\_IoT\_Role'. Key details shown include:

- Role ARN:** arn:aws:iam::089742002813:role/service-role/Lambda\_IoT\_Role
- Instance Profile ARN(s):** None listed.
- Path:** /service-role/
- Creation Time:** 2016-12-21 13:53 PST

Below the summary, there are tabs for 'Permissions', 'Trust Relationships', 'Access Advisor', and 'Revoke Sessions'. The 'Permissions' tab is selected, showing a table titled 'Managed Policies' with two entries:

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 copyright information: © 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Select "AWSIoTFullAccess" and click Attach Policy:

The screenshot shows the 'Attach Policy' dialog box. On the left, a sidebar menu is open under the 'Roles' section, listing 'Policies', 'Identity providers', 'Account settings', 'Credential report', and 'Encryption keys'. The main content area displays the 'Attach Policy' dialog with the following interface elements:

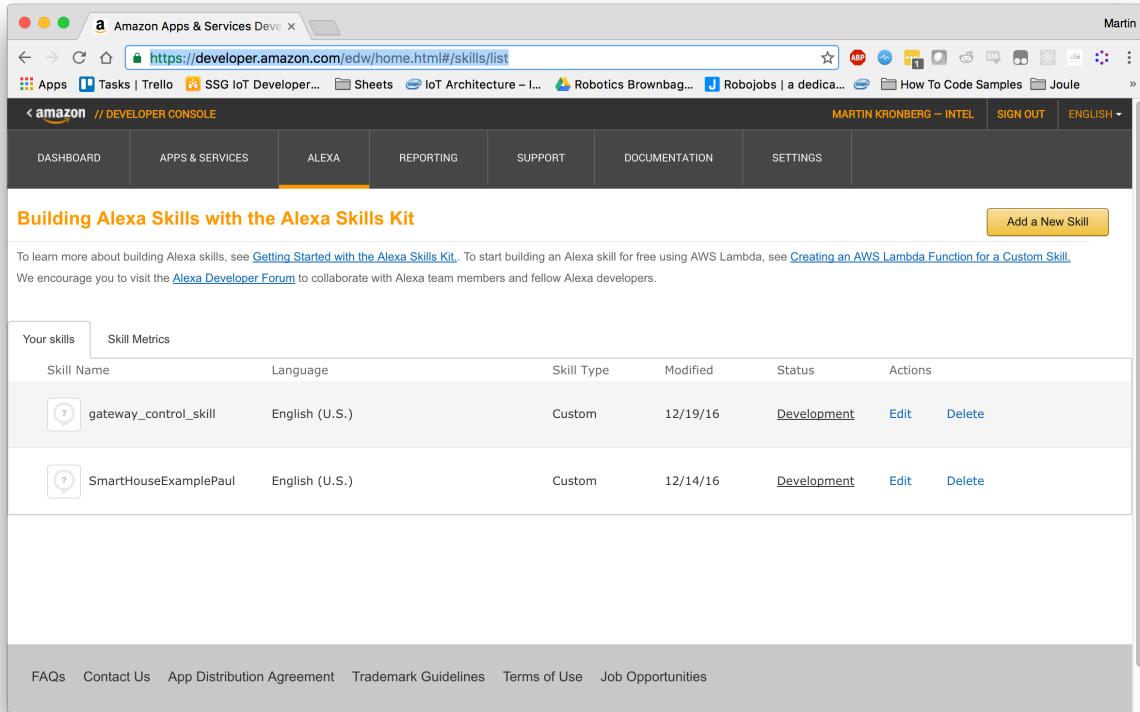
- Title:** Attach Policy
- Sub-instruction:** Select one or more policies to attach. Each role can have up to 10 policies attached.
- Filter:** Policy Type (set to AWS) and a search bar showing 'Showing 98 results'.
- Table:** A list of policies with columns: Policy Name, Attached Entities, Creation Time, and Edited Time. The policy 'AWSIoTFullAccess' is selected (indicated by a checked checkbox).
- Buttons:** 'Cancel' and 'Attach Policy'.

	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"/>	AWSServiceCatalog...	0	2016-08-01 18:35 PST	2016-08-01 18:35 PST
<input type="checkbox"/>	AWSAgentlessDiscovery...	0	2016-05-11 14:38 PST	2016-05-11 14:38 PST
<input type="checkbox"/>	AWSApplicationDiscover...	0	2016-05-11 14:38 PST	2016-05-11 14:38 PST

## 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#/skills/list> and select “Get Started” under Alexa Skills Kit



The screenshot shows the Alexa Developer Console interface. At the top, there's a navigation bar with links like 'DASHBOARD', 'APPS & SERVICES', 'ALEXA' (which is highlighted in orange), 'REPORTING', 'SUPPORT', 'DOCUMENTATION', and 'SETTINGS'. Below the navigation bar, a banner reads 'Building Alexa Skills with the Alexa Skills Kit' and features a 'Add a New Skill' button. 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 shows 'gateway\_control\_skill' in English (U.S.) as a Custom skill last modified on 12/19/16 with a status of 'Development'. The second row shows 'SmartHouseExamplePaul' in English (U.S.) as a Custom skill last modified on 12/14/16 with a status of 'Development'. Each row has 'Edit' and 'Delete' buttons in the 'Actions' column. At the bottom of the page, there are links for 'FAQs', 'Contact Us', 'App Distribution Agreement', 'Trademark Guidelines', 'Terms of Use', and 'Job Opportunities'.

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

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.

The screenshot shows the 'Skill Information' step of the Alexa skill creation wizard. The 'Skill Type' section is expanded, showing 'Custom Interaction Model' selected. Other options like 'Smart Home Skill API' and 'Flash Briefing Skill API' are available. The 'Language' field is set to 'English (U.S.)'. The 'Name' field contains 'Gateway Skill'. The 'Invocation Name' field contains 'Gateway'. Under 'Global Fields', the 'Audio Player' section is shown with 'No' selected. At the bottom are 'Save' and 'Next' buttons.

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”.

The screenshot shows the 'Intent Schema' section of the Alexa developer console. It displays a JSON code block for defining intents:

```

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

```

Below this, the 'Custom Slot Types' section is shown, with a table mapping 'Device\_States' to its values.

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 a single entry:

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

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 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.

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.

The screenshot shows the 'Endpoint' configuration step in the Alexa Skill Development Kit (EDW). The 'Service Endpoint Type' dropdown is set to 'AWS Lambda ARN (Amazon Resource Name)' (selected), and the 'Region' dropdown is set to 'North America'. Other options like 'HTTPS' and 'Test' are also visible.

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 screenshot shows the AWS Lambda Management Console with the URL [https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/Test\\_Function?tab=code](https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/Test_Function?tab=code). The page displays the configuration for a Lambda function named "Test\_Function". The "Code" tab is selected. A message states: "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." Below this, there is a "Code entry type" dropdown set to "Upload a .ZIP file" and a "Function package" upload button. A note says: "For files larger than 10 MB, consider uploading via S3." There is also an "Enable encryption helpers" checkbox and an "Environment variables" section with a key-value pair. The ARN of the function is listed at the top right.

Click next to go to test your new skill:

The screenshot shows the Amazon Appstore Dev Portal with the URL <https://developer.amazon.com/edw/home.html#/skill/amzn1.ask.skill.899460ba-0a43-41b7-8bf8-5a8...>. The page is titled "Defining the Voice Interface". It shows a Lambda Request and Lambda Response. The Lambda Request is a JSON object with session, application, user, and request details. The Lambda Response is a JSON object with version, response, and sessionAttributes. Below the responses are "Listen" and "Next" buttons. At the bottom, there are links for FAQs, Contact Us, App Distribution Agreement, Trademark Guidelines, Terms of Use, and Job Opportunities.

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”