Lab: Add Voice Control Feature to AWS IoT Solutions

Overview	AWS IoT Device Shadow provides the ability to connect devices with your apps so that you can control the devices in your app. And Alexa Voice Service (AVS) enables you to add voice-powered experiences to your connected devices. In this lab, you will learn how to add voice control features to your AWS IoT Solutions.

AWS IoT

Objectives	After completing this lab, you will be able to: • Create a custom Alexa Skill Kit • Create a Lambda function to integrate with AWS IoT Service Devcie Shadow

Pre-requisites	This lab requires: • AWS account • Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat). • For Microsoft Windows users: Administrator access to the computer. • An Internet browser such as Chrome, Firefox. With camera support would be prefer. • FileZilla – For copying files to the device. (https://filezilla-project.org/)

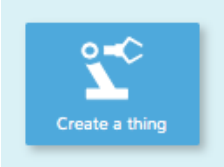
Duration	40 to 60 Minutes

v.1.0.3.0830

Task 1: Create a device in AWS IoT Service

Overview	To connect a device to AWS IoT, we recommend you first create a device in the thing registry. This registry allows you to keep a record of all of the devices that are connected to your AWS IoT account.  AWS IoT

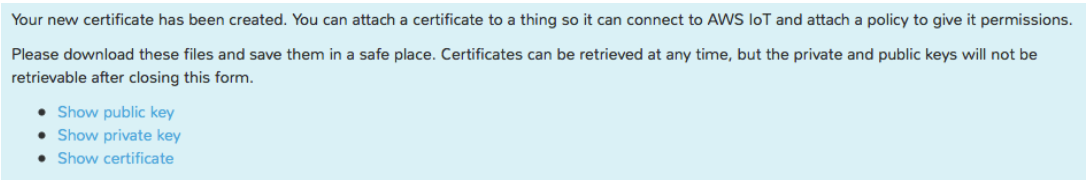
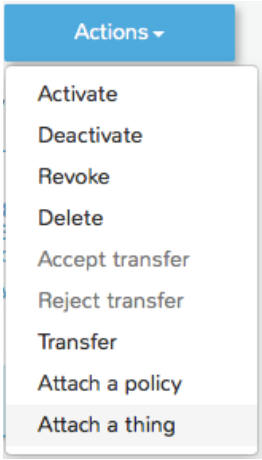
Task 1-1: Create a Device in the Thing Registry	
Overview	In this task you will create a Device in the Thing Registry for further usage in this lab..

Step	Instruction
1.1.1	Sign in AWS Console, make sure you are using Asia Pacific (Tokyo) region, and the select AWS IoT from the menu: 
1.1.2	On the Resources page, select **+ Create a resource** button， and then click on the **Create a thing** item： 

1.1.3	Enter “ **myLight**” in the name column，and click **Create** button。
1.1.4	Now click **View thing** button and you can see the device related information on the right-side panel including REST API end poinr and MQTT Topic….etc.
1.1.5	Click on **Update Shadow** in the right panel.
1.1.6	Update the Shadow state as below: **{** **"desired": {},** **"reported": {"light":0}** **}**
1.1.7	Click on the **Update shadow** button, and you will see the new Shadow state and Shadow metadata in the Details page:
1.1.8	On the Resources page, select **x Close create panel** button. Finish create a device in the thing registry.

Task 1-2: Creating and Activate an AWS IoT Device Certificate and Policy	
Overview	Communication between your AWS IoT button and AWS IoT is protected through the use of X.509 certificates. AWS IoT can generate a certificate for you or you can use your own X.509 certificate. AWS IoT policies are used to authorize your button to perform AWS IoT operations, such as subscribing or publishing to MQTT topics. Your button will present its certificate when sending messages to AWS IoT. To allow your button to perform AWS IoT operations, you must create an AWS IoT policy and attach it to your device certificate. In this task we will use AWS IoT to generate an X.509 certificate and activate it for you. Also we will create an AWS IoT policy and attach it to the device certificate.

Step	Instruction
1.2.1	On the Resources page, select **+ Create a resource → Create a certificate** : 

1.2.2	Click **1-Click certificate create** button, and AWS IoT will create the keys and certificate for you：  **[Note] Please DO NOT close this page before finish downloading the keys and certificate.**
1.2.3	Create an **IoTCerts** folder in your filesystem. On the **Resources** page, choose the **Download private key** and **Download certificate** links, and save the private key and certificate to your computer.
1.2.4	On the Resources page, select **x Close create panel** button.
1.2.5	Select the check box on the certificate which we just created, and from the **Actions** menu, choose **Attach a thing**. 
1.2.6	In the **Confirm** windows, type **myLight** in the **Thing name** column, and then select **Attach** button.
1.2.7	Repeat 1.2.5, select **Actions**, and then select **Activate** from the menu. AWS IoT now can use this certificate to identify this deveice.

1.2.8	On the Resources page, select **+ Create a resource** → **Create a policy** : 
1.2.9	In the **Create a policy** section, type **myFirstThing-policy** in the **Name** column. From the **Action** menu, choose **iot:***. In the **Resource** field, type * to allow access to all AWS IoT resources. Select the Allow check box, choose Add statement, and then choose Create.
1.2.10	Now we need to attach this policy to the device certificate. Repeat 1.2.5, select **Actions**, and then select **Attach a policy** from the menu.
1.2.11	In the **Confirm** windows, type **myFirstThing-policy** in the **Policy name** column, and then select **Attach** button.

Task 2: Create an Alexa Skill Kit

Overview	AWS IoT Service can integrate with your applications to let users to manage their devices. In this task, you will learn how to create an Alexa Skill Kit to control your device via AWS IoT.

Task 2-1: Create a Lambda function for Alexa Skill Kit

Overview	The easiest way to build the cloud-based service for a custom Alexa skill is by using AWS Lambda, so there is no need to provision or continuously run servers. AWS Lambda supports JavaScript and you upload the code for your Alexa skill to a Lambda function and Lambda does the rest, executing it in response to Alexa voice interactions and automatically managing the compute resources for you.

Step	Instruction
2.1.1	Open AWS Lambda Console, and click on **Create a Lambda function → Skip**.

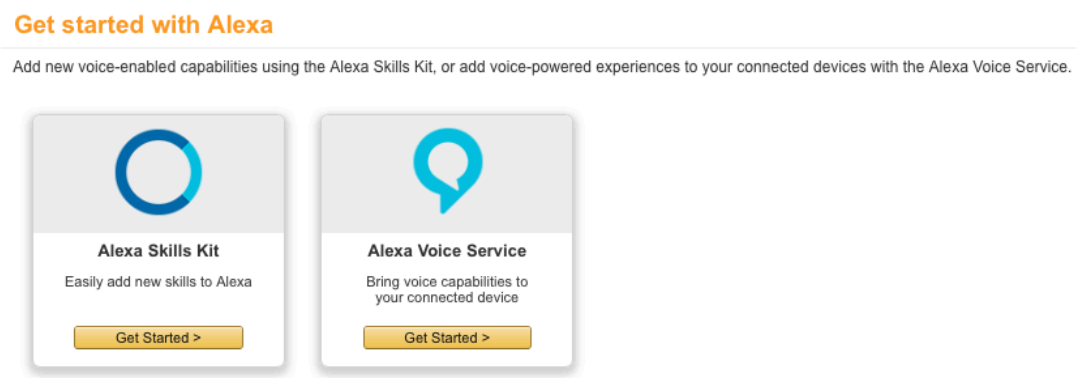
2.1.2	In the Configure function page, named the function **skill-light-control**, and add some description. Make sure the Runtime is **Node** and **Edit code inline** is selected as well.
2.1.3	Please download the Alexa Skill Lambda template from: **http://bit.ly/Alexa-Mini-Lambda**. Open the file downloaded and copy/paste the contents of the file to Lambda function code.
2.1.4	Add below code at the beginning of the Lambda code: **//Environment Configuration** **var config = {};** **config.IOT_BROKER_ENDPOINT = "[AWS IoT endpoint of your thing]".toLowerCase();** **config.IOT_BROKER_REGION = "us-east-1";** **config.IOT_THING_NAME = "myLight";** **var AWS = require('aws-sdk');** **AWS.config.region = config.IOT_BROKER_REGION;** **var iotData = new AWS.IotData({endpoint: config.IOT_BROKER_ENDPOINT});**

2.1.5	Add below code to the end of the Lambda code: **function lightUp (intent, session, callback) {** **var repromptText = null;** **var sessionAttributes = {};** **var shouldEndSession = true;** **var speechOutput = "";** **var payloadObj={ "state":** **{ "desired":** **{"light":1}** **}** **};** **//Prepare the parameters of the update call** **var paramsUpdate = {** **"thingName" : config.IOT_THING_NAME,** **"payload" : JSON.stringify(payloadObj)** **};** **//Update Device Shadow** **iotData.updateThingShadow(paramsUpdate, function(err, data) {** **if (err){** **//Handle the error here** **}**

	else { **speechOutput = "The light has been turned on!";** **console.log(data);** **callback(sessionAttributes,buildSpeechletResponse(intent.name, speechOutput, repromptText, shouldEndSession));** **}** **});** **}**
2.1.6	Find the onIntent method, modified the code as below: **function onIntent(intentRequest, session, callback) {** **console.log("onIntent requestId=" + intentRequest.requestId** **+ ", sessionId=" + session.sessionId);** **var intent = intentRequest.intent,** **intentName = intentRequest.intent.name;** **lightUp(intent,session,callback)** **}**
2.1.7	In the Lambda function handler and role, select **Create new role → *Basic execution role** in the Role column. In the new prompt window, make sure the new role have the **iot DataAccess** policy.
2.1.8	In the new Windows, select **Allow.**
2.1.9	In the Lambda console, select **Next → Create function**.
2.1.10	After the Lambda function created, select the **Event sources** tab.

2.1.11	Select **Add event source**, and select **Alexa Skills Kit** in the Event source tye. Then click on the **Submit** button.
2.1.12	Copy the ARN of the Lambda function on the right-top corner. The ARN format should be like this: **arn:aws:lambda:us-east-1:XXXXXXXXXX:function:skill-light-control**

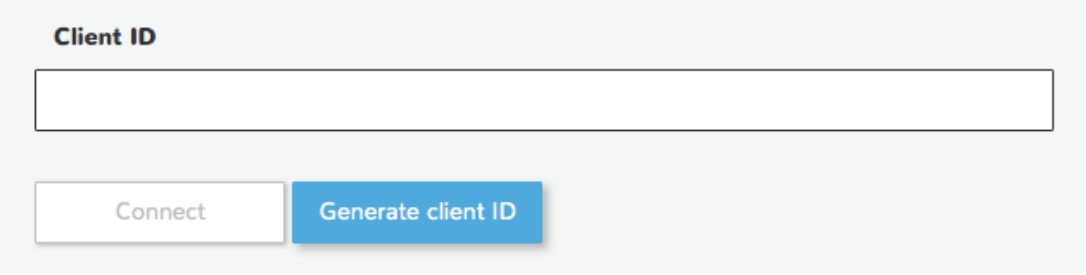
Task 2-2: Create an Alexa Skills Kit	
Overview	In this task, we will create a new Alexa Custom Skills Kit to trigger the Lambda function

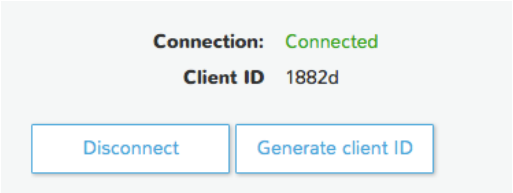
Step	**Instruction**
2.2.1	Sign up and login Amazon Developer Portal (https://developer.amazon.com) with your Amazon account.
2.2.2	Click on **Alexa**, and then click on **Get Started** of Alexa Skill Kit. 
2.2.3	Select Add a New Skill.
2.2.4	In the Create a New Alexa Skill page, select **Custom Interaction Model** as the skill type. Enter **Light Control** in the name field and **my agent** in the invocation name field. Click **Next** when finished.

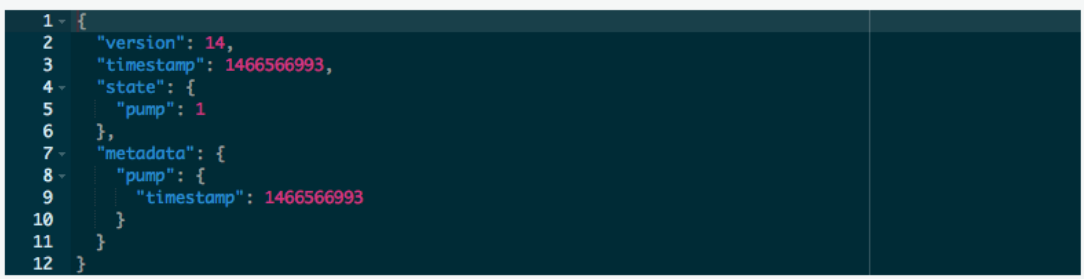
2.2.5	In the Interaction Model page, enter below script into Intent Schema: **{** **"intents": [** **{** **"intent": "LightIntent"** **}** **]** **}**
2.2.6	Enter below script into Sample Utterances and the click **Next**: **LightIntent turn on the light.**
2.2.7	In the Configuration page, select Lambda ARN as the endpoint and paste the ARN which you copied in the step 2.1.12. It should be similar as below: **arn:aws:lambda:us-east-1:xxxxxxxxxx:function: skill-light-control**
2.2.8	Select **No** on the account linking, and then click **Next**.

2.2.9	Now we can test the voice control feature in the Test page. In the Service Simulator section, enter "**Alexa, ask my agent to turn on the light.**" Into the Enter Utterance field and click on the **Ask Light Control** button. Then you will found "The light has been turned on!" output speech message in the Lambda Response as below:
	
	Lambda Response
	
	`{`
	`  "version": "1.0",`
	`  "response": {`
	`  "outputSpeech": {`
	`  "type": "PlainText",`
	`  "text": "The light has been turned on!"`
	`  },`
	`  "card": {`
	`  "content": "SessionSpeechlet - The light ha`
	`  "title": "SessionSpeechlet - LightIntent",`
	`  "type": "Simple"`
	`  },`

Task 2-3: View Device MQTT Messages with the AWS IoT MQTT Client	
Overview	Before connected to a real device, you can use AWS IoT MQTT client to better understand the MQTT messages sent by a device. Devices publish MQTT messages on topics. You can use the AWS IoT MQTT client to subscribe to these topics to see the content of these messages.

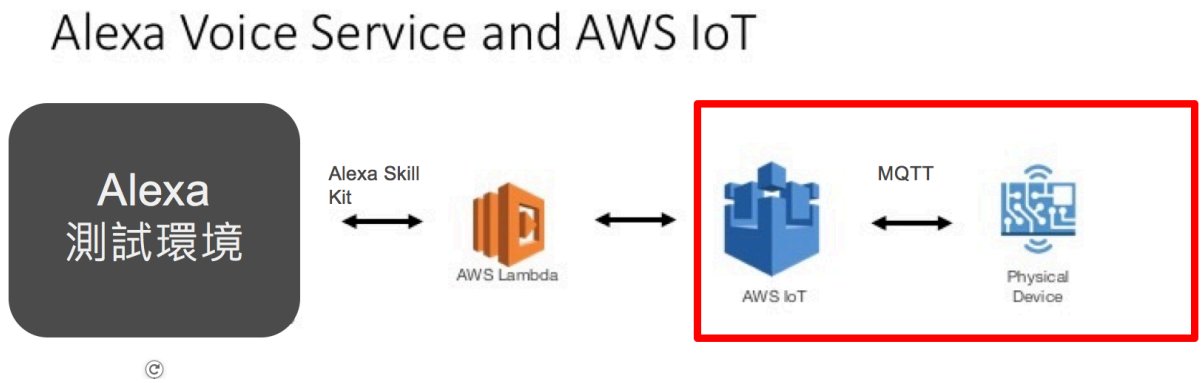
Step	Instruction
2.3.1.	In the AWS IoT console, choose **MQTT Client**.
2.3.2.	Select **Divice Gateway connection**: 
2.3.3.	Choose Generate client ID to create a client id which will be included in each AWS IoT message: 

2.3.4.	Click on **Connect** button and connect to AWS IoT service. The Connect status will be “Connected” if success: 
2.3.5.	Then subscribe to the topic on which your thing publishes. Choose Subscribe to topic, and in Subscription topic, type **$aws/things/myLight/shadow/update/delta**, and then choose **Subscribe**.

2.3.6.	Execute Step 2.2.9 again, then MOTT Client will receive the delta message as below:
	 
	{
	"version": 14,
	"timestamp": 1466566993,
	"state": {
	"pump": 1
	},
	"metadata": {
	"pump": {
	"timestamp": 1466566993
	}
	}
	}

Task 3: Connect your device to AWS IoT (Intel Edison)

Overview	After test the voice features, now let's see how to setup and connect your device to AWS IoT. In this task, we will use JavaScript to write code to turn the LED light on.



Task 3-1: Setup your device (Intel Edison)

Overview	Before starting to write code, we need to setup the device first. In this lab, we had help to pre-configure the device with the latest firmware and pre-install necessary JavaScript Libraries, but you still need to connect the LED light and setup wifi connection.

Step	Instruction

3.1.1	Check your components. We will need: ● Intel Edison with Arduino Expantion Board ● Breadboard ● LED Light ● 2 USB Cables ● Connecting Wires
3.1.2	Once you have all of the components, connect the LED light to the D13 pin and then connect both USB cables to your computer for developement.
3.1.3	Now we need to setup a serial terminal to the device. ● For Windows, ■ Please install USB Drivers: https://software.intel.com/zh-cn/installing-drivers-for-intel-edison-board-with-windows ■ Follow this article to download and set up PuTTY. ● For MacOS, ■ Follow this article to set up serial communication: https://software.intel.com/zh-cn/setting-up-serial-terminal-on-system-with-mac-os-x
3.1.4	Login to your board by the credentials below: Name: **root** Password: **password**
3.1.5	Configure WiFi with your classroom. Enter below command and provide necessary information in the following steps: **configure_edison –wifi**
3.1.6	Open FileZilla tool and connect to your Intel Edison by the IP address which you got from the previous step.
3.1.7	Copy the files which you downloaded in the Step 1.2.3 from the **IoTCerts** folder to your device.
3.1.8	Download the sample script **CloudKata.js** from **http://bit.ly/cloudkata-iotlab01** and open it within your text editor.
3.1.9	Go to line 9 & 10, modify the certification filename and the private key file name to match the files which you uploaded in the Step 3.1.7.

3.1.10	Save the file and upload it to your Intel Edison device via FileZilla tool.
3.1.11	Open the serial terminal window and enter below command: **node CloudKata.js**
3.1.12	Execute Step 2.2.9 again. Now you should see the LED is turned on.