

Lab 1: Setting up an Amazon Cognito Identity Pool

Overview	<p>In this lab you will setup an Amazon Cognito Identity Pool and Register Facebook as a third-party authentication provider to allow users to use your application. You will also set up your own Identity Provider which we will implement later in the Bootcamp.</p> <p>When the Amazon Cognito Identity Pool is set up, we will test Amazon Cognito Sync using a test web site</p>
-----------------	--

Objectives	<p>After completing this lab, you will be able to:</p> <ul style="list-style-type: none">• Create an Amazon Cognito Identity Pool to allow both Authenticated and UnAuthenticated users for your application.• Use a Facebook Social Application as your third-party authentication provider.• Associate the Social Application with your Identity Pool.• Access shared user data in the Amazon Cognito Sync Store via the AWS Management Console.• Create a simple static site using S3 buckets.• Access shared user data in the Amazon Cognito Sync Store from the above website.
-------------------	--

Pre-requisites	<p>This lab requires:</p> <ul style="list-style-type: none">• Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">○ Note The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.• For Microsoft Windows users: Administrator access to the
-----------------------	--

	<p>computer.</p> <ul style="list-style-type: none">• An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).
--	--

Duration	30 minutes
-----------------	------------

Task 1: Creating an Amazon Cognito Identity Pool

Overview	Create an Amazon Cognito Identity Pool that allows for both authenticated and unauthenticated users to access your application. In this case users will use Facebook as a third-party authentication provider.
-----------------	--

Task 1-1: Create an Amazon Cognito Identity Pool

Overview	In this section you will create an Amazon Cognito Identity Pool. An identity pool is a store of user identity data specific to your account. Using Amazon Cognito Sync, you can retrieve data across client platforms, devices, and operating systems, so that if a user starts using your app on a phone and later switches to a tablet, the persisted app information is still available for that user.
-----------------	---

Step	Instruction
------	-------------

1.1.1

To streamline the Bootcamp, we have prepared some files that are almost complete, but need you to edit to apply some detail specific to your Qwiklabs account before you can use them. These files are bundled into a ZIP file and stored in the AWS Cloud.

In this first step, you will retrieve the bundle, and unzip the files to a location on your computer.

You can download the bundle by clicking this link or pasting it into a web browser:

<http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/Desktop.zip>

If you are on a Mac you can alternatively download with the following commands:

```
$ cd ~/Desktop/  
$ curl -O "http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/Desktop.zip"
```

Download this bundle and expand the ZIP file on your computer. We will refer back to this ZIP file during the Lab guides, so remember where you put the files! We will need to refer to these files a few times during our Bootcamp.

- 1.1.2 Navigate to the **bootcamp.config** file that you downloaded as part of the Desktop.zip bundle in the previous step, and unpacked on to your computer.
- Open the **bootcamp.config** file in a text editor. You can choose whatever text editor you are comfortable with, but you should not use a rich text editor like MSWord.
- At the top of the file, you will find two configuration sections that you need to update:

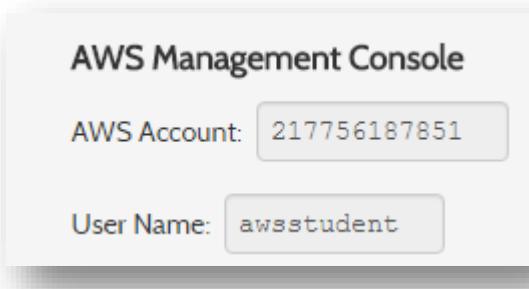
```
///////////////////////////////
//  
// Replace the "XXXXXXXXXXXX" in the  
// line below, to be your Qwiklabs Account Id  
// without any dashes  
//  
// It will look something like this "217756187851"  
//  
///////////////////////////////  
  
this.AWS_ACCOUNT_ID      = "XXXXXXXXXXXXXX";  
  
///////////////////////////////
//  
// Replace the "XXXXXXXXXXXX" in the  
// line below, to be the Cognito Identity pool Id  
// exactly as it is shown in the AWS Console  
//  
// It will look something like this "us-east-1:223e168c-5ddk  
//  
///////////////////////////////  
  
this.COGNITO_IDENTITY_POOL_ID  = "XXXXXXXXXXXXXX";
```

1.1.3

Provide your AWS Account Id.

Notice the 'XXXXXXXXXXXX' value set for `this.AWS_ACCOUNT_ID`?

Change this value to the AWS Account Id for your Qwiklabs account. You can get the Account Id by viewing the Qwiklabs dashboard, where you started the Qwiklab from. An example of the Account Id is shown below:

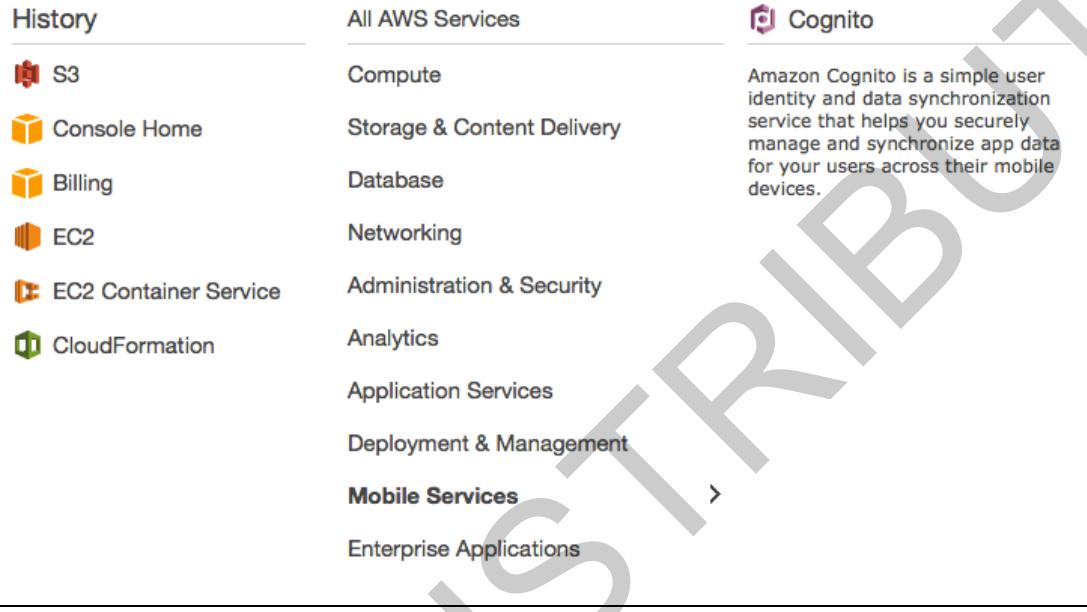


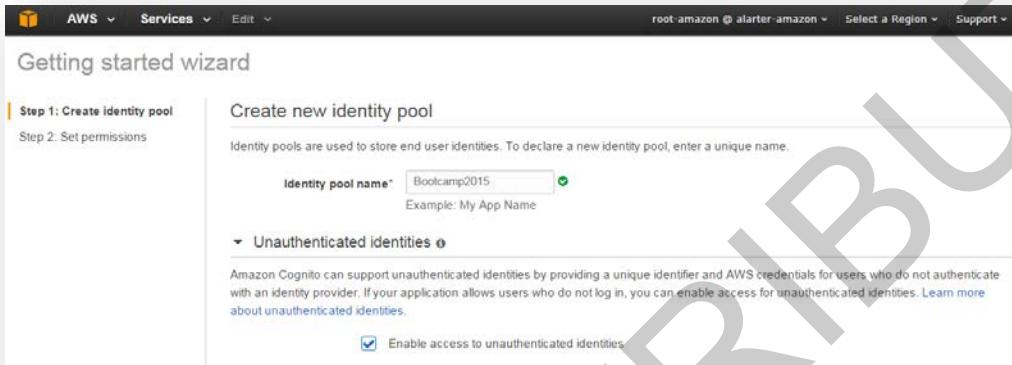
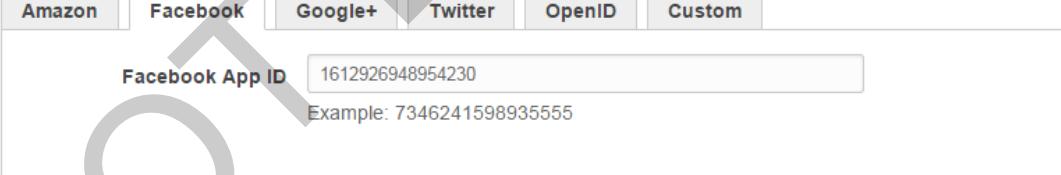
Note: Your account Id will be different! You cannot use the account Id shown above, you must use the account Id for the Qwiklabs account you are using for the Bootcamp!

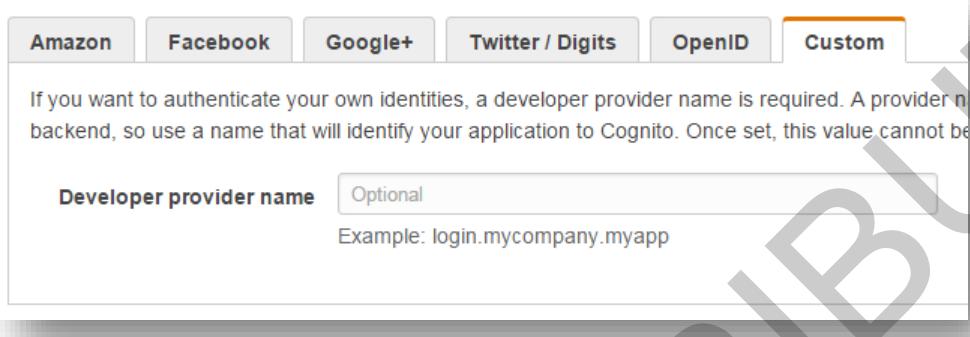
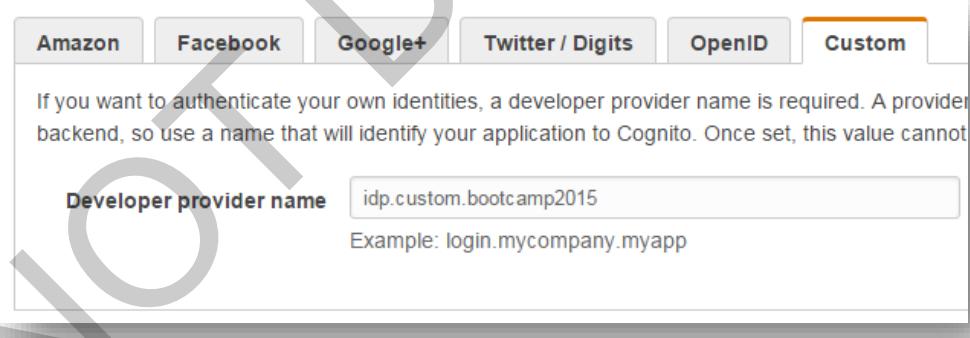
After you have entered your account Id, the line should look something like this:

```
this.AWS_ACCOUNT_ID = "217756187851";
```

We will come back to this file in a moment, after we have created our Amazon Cognito Identity Pool.

1.1.4	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services -> Mobile Services -> Cognito</p>  <p>The screenshot shows the AWS Management Console navigation bar. On the left is a 'History' sidebar with icons for S3, Console Home, Billing, EC2, EC2 Container Service, and CloudFormation. To the right is a 'All AWS Services' menu with categories: Compute, Storage & Content Delivery, Database, Networking, Administration & Security, Analytics, Application Services, Deployment & Management, Mobile Services (which is selected and highlighted in blue), and Enterprise Applications. On the far right is a 'Cognito' service card with its icon, name, and a brief description: 'Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices.'</p>
1.1.5	<p>Click the Get Started button.</p>  <p>The screenshot shows the Amazon Cognito landing page. It features the Cognito logo and the heading 'Amazon Cognito'. Below the heading is a description: 'Amazon Cognito helps you securely store, manage, and sync identities and data across multiple devices, platforms, and applications.' At the bottom is a prominent blue 'Get started' button.</p>

1.1.6	<p>Give the Identity Pool the name Bootcamp2015 and check the Enable access to unauthenticated identities checkbox.</p> 
1.1.7	<p>Scroll down and open the Authentication Providers section. Choose Facebook, and paste the following Facebook App ID: 1612926948954230</p> 

1.1.8	<p>Now click the Custom tab.</p>  <p>We will be implementing our own custom Developer Authentication Provider later, so we will set up our Developer Provider Name now. In the text field, enter</p> <pre>idp.custom.bootcamp2015</pre> <p>When you have finished, the panel should look like this:</p> 
1.1.9	<p>Click Create Pool in the bottom right.</p>
1.1.10	<p>In the next screen you will be asked to create two Policy Documents for two roles that Amazon Cognito uses. One role is for the users who have authenticated themselves via Facebook, and the other is for users who have not authenticated themselves. Amazon Cognito will use these roles to identify which AWS resources the user has access to.</p>

1.1.11

Click the **View Details** button.

Your Cognito identities require access to your resources

Assigning a role to your application end users helps you restrict access to your AI
By default, Amazon Cognito creates a new role with limited permissions - end user

▶ **View Details**

Set the name of **Authenticated** role to:

Bootcamp2015-Cognito-Authenticated

Set the name of **Unauthenticated** role to:

Bootcamp2015-Cognito-UnAuthenticated

▼ Hide Details

Role Summary	
Role Description	Your authenticated identities would like access to Cognito.
IAM Role	Create a new IAM Role ▾
Role Name	Bootcamp2015-Cognito-Authenticated

▶ View Policy Document

Role Summary	
Role Description	Your unauthenticated identities would like access to Cognito.
IAM Role	Create a new IAM Role ▾
Role Name	Bootcamp2015-Cognito-UnAuthenticated

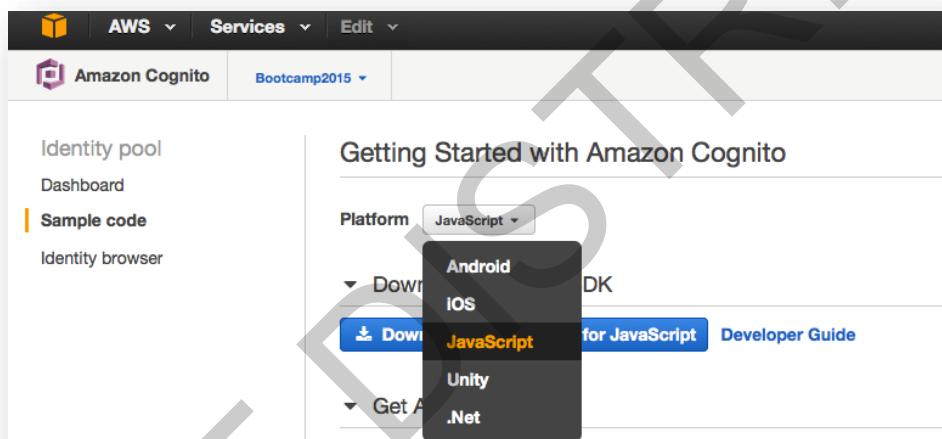
▶ View Policy Document

- 1.1.12 Click the **Allow** button in the bottom right to create the roles to continue.

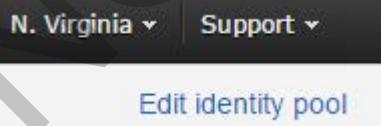


You will be directed to the **Getting Started with Amazon Cognito** page.

We do not need to use these code samples in our Bootcamp today.



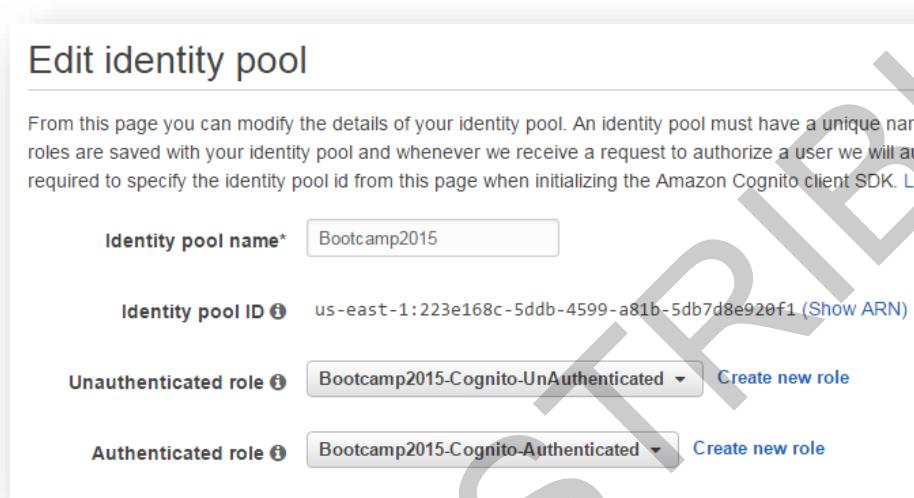
- 1.1.13 In the **top right** of the page, click the **Edit Identity Pool** link:



1.1.14

The page will refresh

You will now see the Edit Identity Pool page. The Identity Pool Id is shown in the list of values:



Copy the value of the Identity Pool ID into your clipboard

You need the entire string, including the 'us-east-1:' section. So, your full Identity Pool Id that you need to copy into your clipboard will look like this:

us-east-1:223e168c-5ddb-4599-a81b-5db7d8e920f1

But of course, your Id will be different to that shown above!

*Note that on Safari it may not display as above. If you notice an @ sign try using a different browser or asking an instructor for assistance.

1.1.15	<p>With the Identity Pool ID in your clipboard, we will now update the <code>bootcamp.config</code> file again to add the value in.</p> <p>Paste in your Amazon Cognito Identity Pool Id</p> <p>Notice the 'XXXXXXXXXXXXXX' value set for <code>this.COGNITO_IDENTITY_POOL_ID</code> in the text file:</p> <pre>/////////////////////////////// // Replace the "XXXXXXXXXXXXXX" in the // line below, to be the Cognito Identity pool Id // exactly as it is shown in the AWS Console // // It will look something like this "us-east-1:223e168c-5ddb-4599-a81b-5db7d8e920f1" // /////////////////////////////// this.COGNITO_IDENTITY_POOL_ID = "XXXXXXXXXXXXXX";</pre> <p>Update the configuration file like you did before for the Account Id, but this time, paste the value for <code>this.COGNITO_IDENTITY_POOL_ID</code> with the value in your clipboard.</p> <pre>this.COGNITO_IDENTITY_POOL_ID = "us-east-1:223e168c-5ddb-4599-a81b-5db7d8e920f1";</pre> <p>When you have finished, the value should look similar to the above, <i>but note that your Identity Pool Id will be different.</i></p>
1.1.16	<p>Save the text file on your PC/Mac. In the next steps, we will upload this edited configuration file to S3 so we can test Amazon Cognito on a web page.</p>

Task 2: Creating an S3 Bucket and Uploading the configuration file

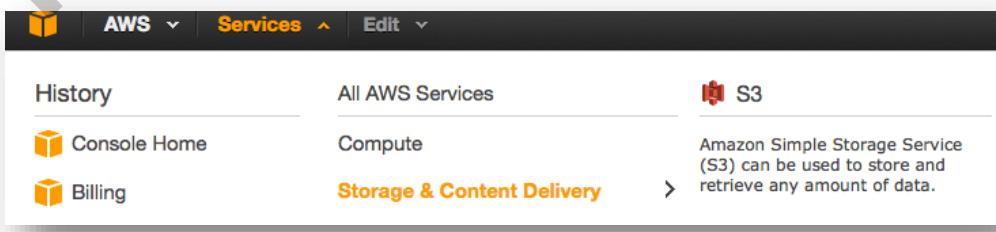
Overview

We will use a sample website to validate your Amazon Cognito settings to make sure everything is setup properly. This website is already deployed to a shared S3 bucket, but you will need to deploy your own configuration file in your own S3 bucket in order to connect to your Amazon Cognito Identity Pool.

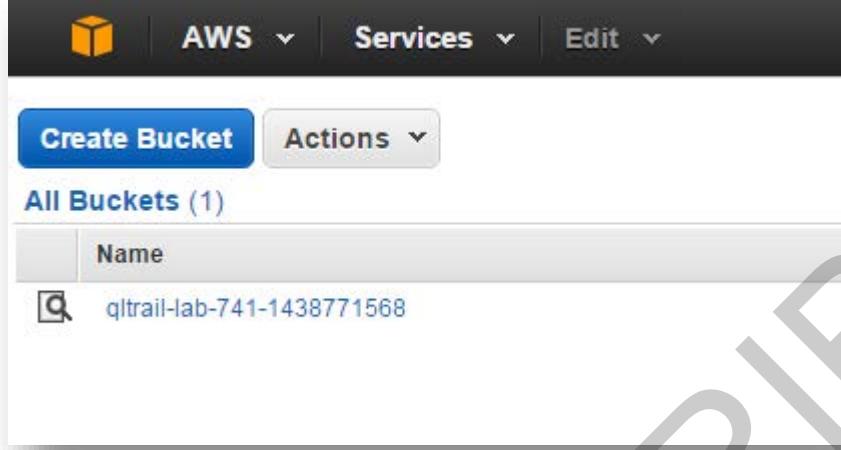
Task 2-1: Create an S3 Bucket to Hold Web Assets

Overview

In this section you will create an S3 bucket to hold a configuration file that will allow the shared web page to connect to your Amazon Cognito Identity Pool and test this part of the Bootcamp.

Step	Instruction
2.1.1	<p>Launch the AWS Management Console from your Qwiklabs environment and click S3 under Storage and Content Delivery from the drop-down list.</p> 

2.1.2



Click **Create Bucket** and enter a bucket name in the following format:
lastname-firstname-bootcamp2015

So if your name is Adam Larter, you would use a name like this:

larter-adam-bootcamp2015

You cannot use this name – you must provide your own!

Choose region as **US-Standard**, then click **Create**.

Make sure all the letters are in lower case

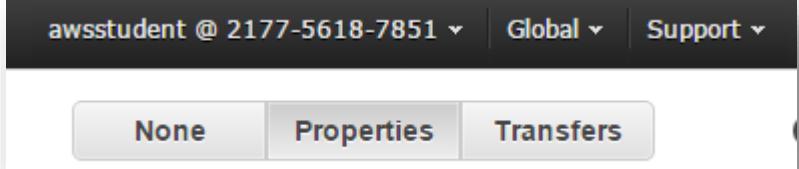
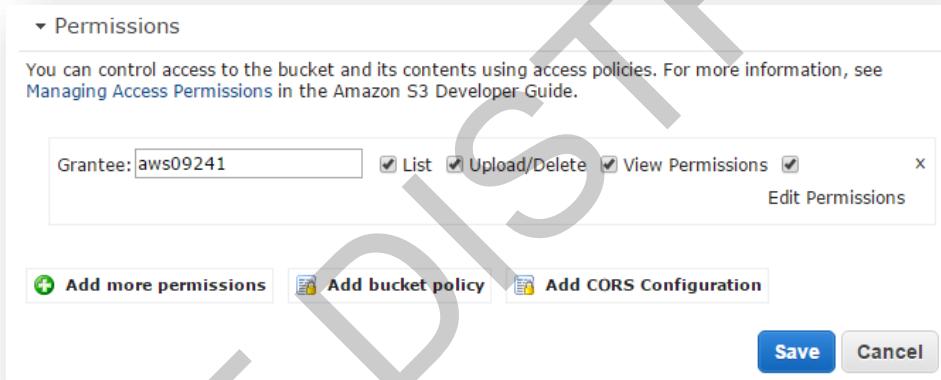
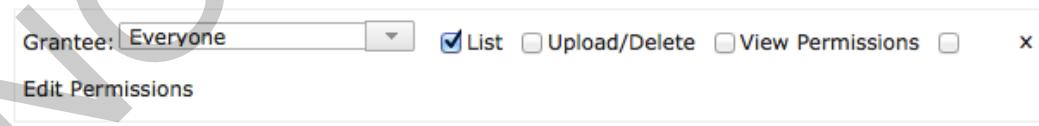
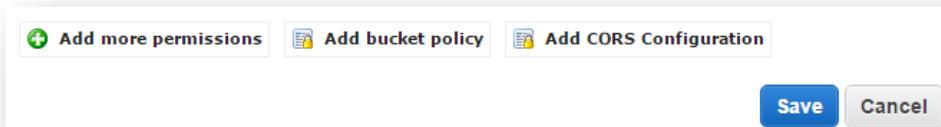
Create a Bucket - Select a Bucket Name and Region

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the [Amazon S3 documentation](#).

Bucket Name:

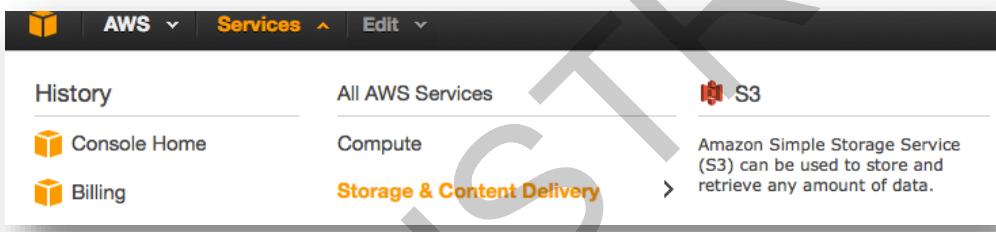
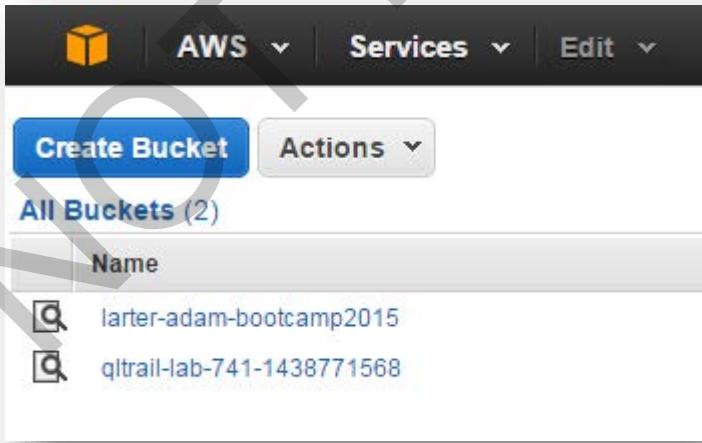
Region:

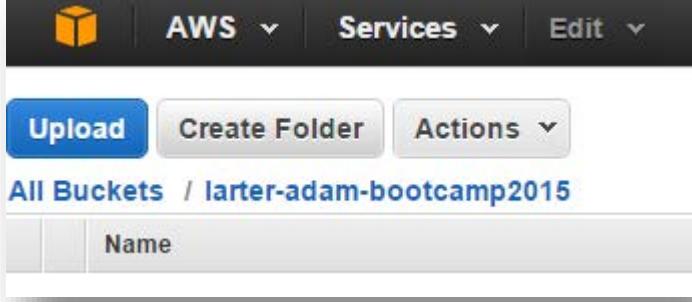
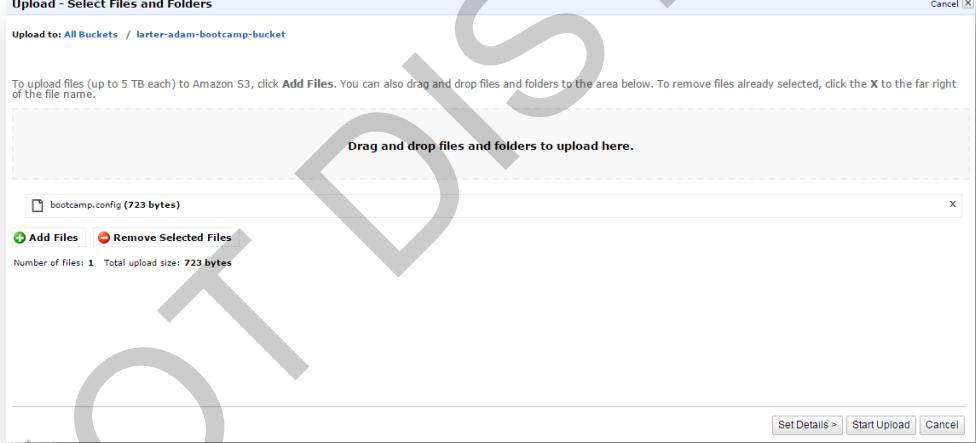
[Set Up Logging >](#) [Create](#)

2.1.3	<p>Click the newly created bucket and click Properties at the top right of the page.</p>  <p>Expand the Permissions tab and click Add more permissions.</p> 
2.1.4	<p>Choose the Grantee as Everyone and then check the List checkbox.</p> 
2.1.5	<p>Then click Save.</p> 

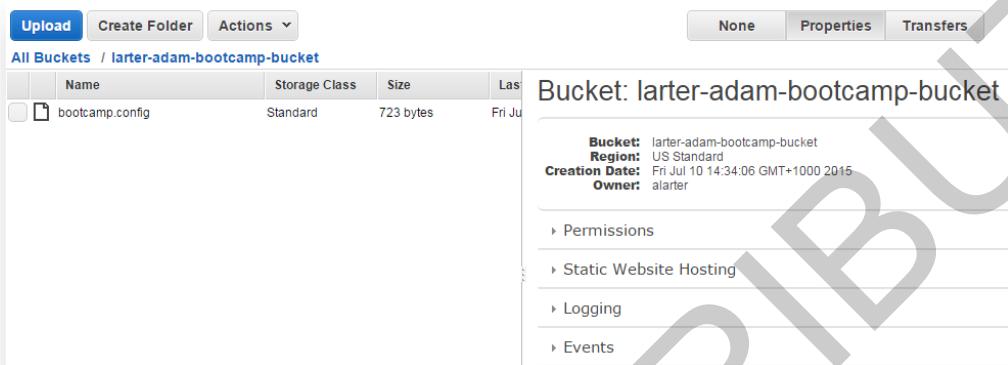
Task 2-2: Upload the config file to the S3 Bucket

Overview In this section we will upload the config file to S3.

Step	Instruction
2.2.1	If you are not already viewing the S3 panel from the AWS console, launch the AWS Management Console from your Qwiklabs environment and click S3 under Storage and Content Delivery from the drop-down list. 
2.2.2	Within S3 choose click on the bucket name you created in step 2.1.2. 

2.2.3	<p>The panel will update and show the upload option</p> 
2.2.4	<p>Click Upload and either use the drag and drop action, or the Add Files option to choose the bootcamp.config file you just edited.</p> 
2.2.5	<p>Click Start Upload.</p>

- 2.2.6 When the file is uploaded click the **Properties** button in the top right corner of the page.



- 2.2.7 Before we access the test website we need to make sure that our bucket contents (the bootcamp.config file) are publicly available. We do this by adding a **Bucket Policy** document to our bucket. You can access this by clicking on the **Properties** button of the bucket.

2.2.8	<p>Expand the Permission tab, click Add Bucket Policy and enter the following policy:</p> <pre>{ "Version": "2012-10-17", "Statement": [{ "Sid": "PublicReadForGetBucketObjects", "Effect": "Allow", "Principal": "*", "Action": ["s3:GetObject"], "Resource": ["arn:aws:s3:::XXXXXXXXXXXX/*"] }] }</pre> <p>NOTE: Replace XXXXXXXX given above with the name of your bucket, eg: <i>larter-adam-bootcamp2015</i></p> <p>So the resource line would become:</p> <pre>"Resource": ["arn:aws:s3:::larter-adam-bootcamp2015/*"]</pre>  <p>2.2.9 Click Save and then click Save again.</p>
-------	---

Task 2-3: Navigate to the Website and Test Amazon Cognito

Overview	You will use a test website to make sure your Amazon Cognito Identity Pool is setup correctly, as well as see Amazon Cognito in action by trying to manipulate some datasets in Amazon Cognito.
-----------------	---

Step	Instruction
2.3.1	<p>Open a web browser and enter the following URL. Note that you will need to replace YOUR_BUCKET_NAME with the name of the bucket you created in section 2.1.2</p> <p><u>http://bootcamp2015-mobile-iot.s3.amazonaws.com/lab1/cognitosync.html?bucket=YOUR_BUCKET_NAME.s3.amazonaws.com</u></p> <p>The webpage will read the query string, extract the value of <i>bucket</i> and then attempt to load the file <i>bootcamp.config</i> from the bucket's root. The file that loads will be your configuration file, with the replaced values. This will then instruct the web page to connect to your account/Amazon Cognito Identity Pool.</p>

2.3.2 When the page loads, you will see the webpage below – if you open the developer panel, you will also see console log output (chrome browser shown here). The screen shot below shows the output when you have previously authenticated using Facebook: that is, if you refresh the page after authentication, it will automatically re-authenticate.

If this is the first time you have hit the page, you will need to authenticate (see the next section)



2.3.3

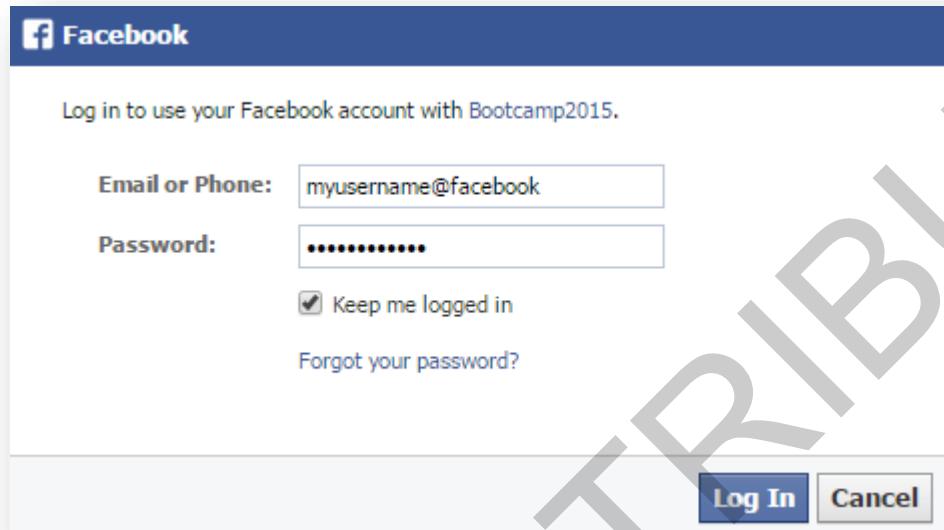
You will need to connect with Facebook. Do so by clicking the **Connect with Facebook** link on the webpage.



[Connect with Facebook](#)

2.3.4

A Facebook login window will appear.



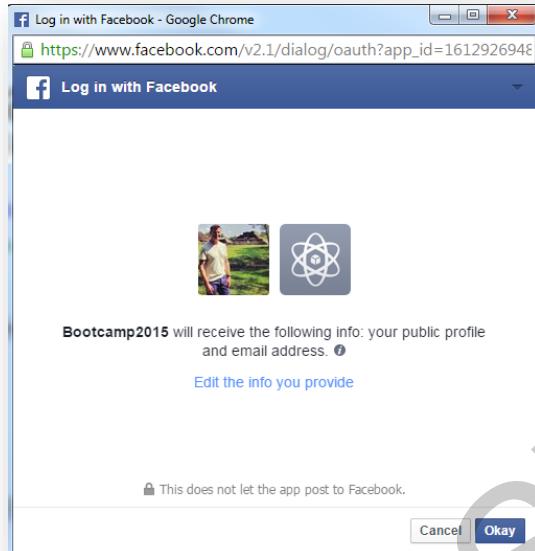
If you have a Facebook account, you can use that. Enter your personal Facebook username and password.

If you do not have a Facebook account, or prefer not to use it, you can use one of our test accounts instead:

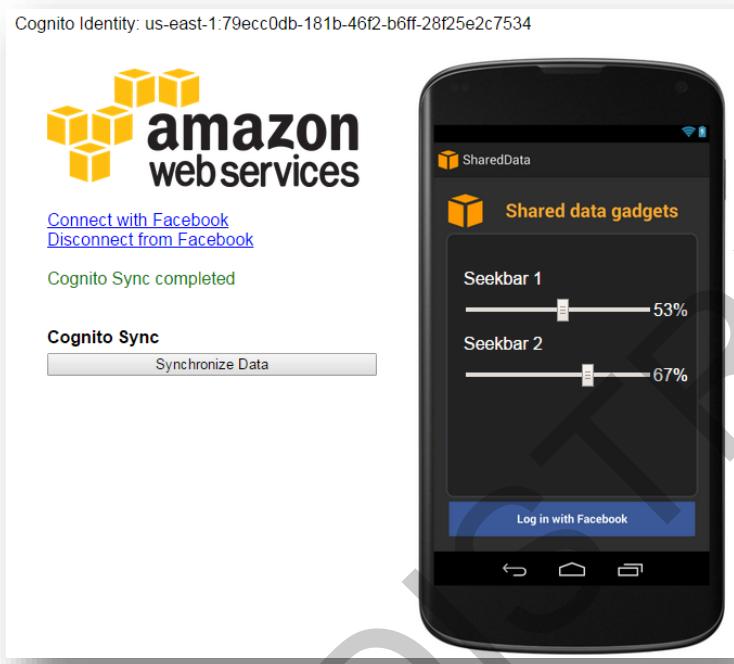
User name: boot_epsobwb_user@tfbnw.net

Password: bootcamp2015

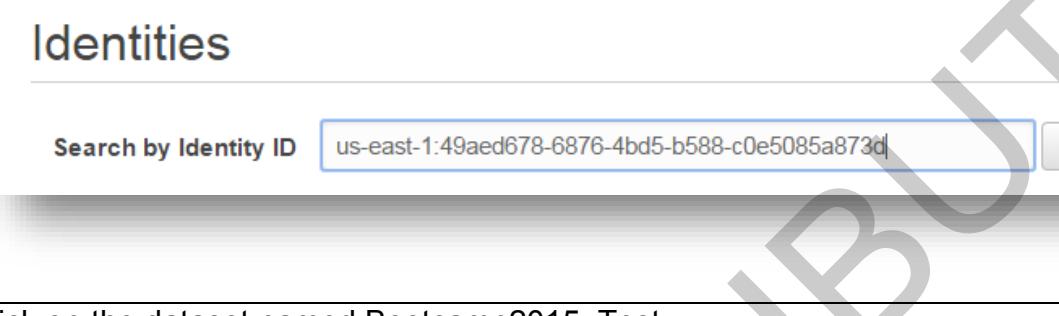
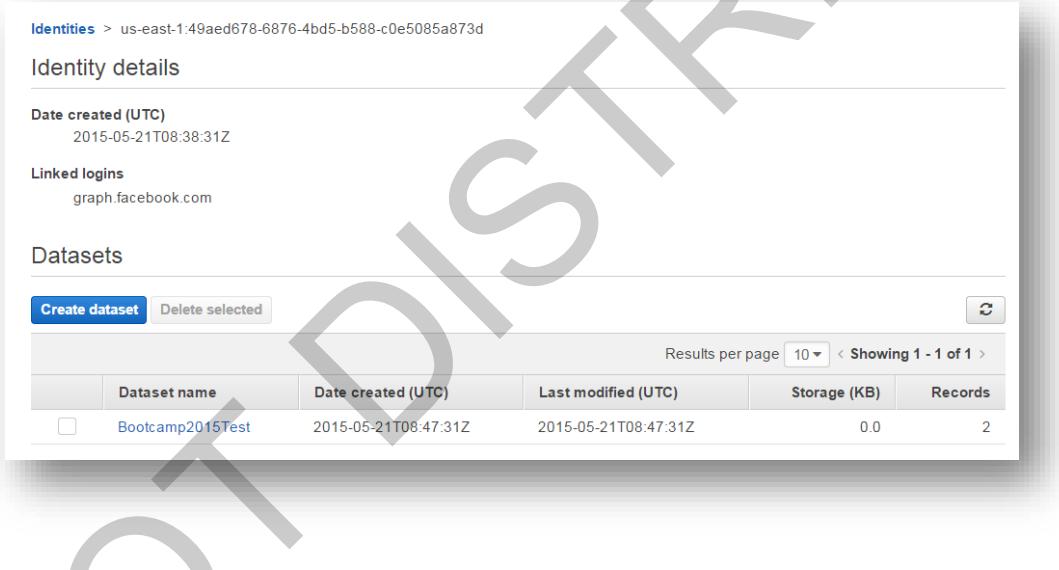
- 2.3.5 On the page click **Okay** to give webpage the access to your Facebook profile

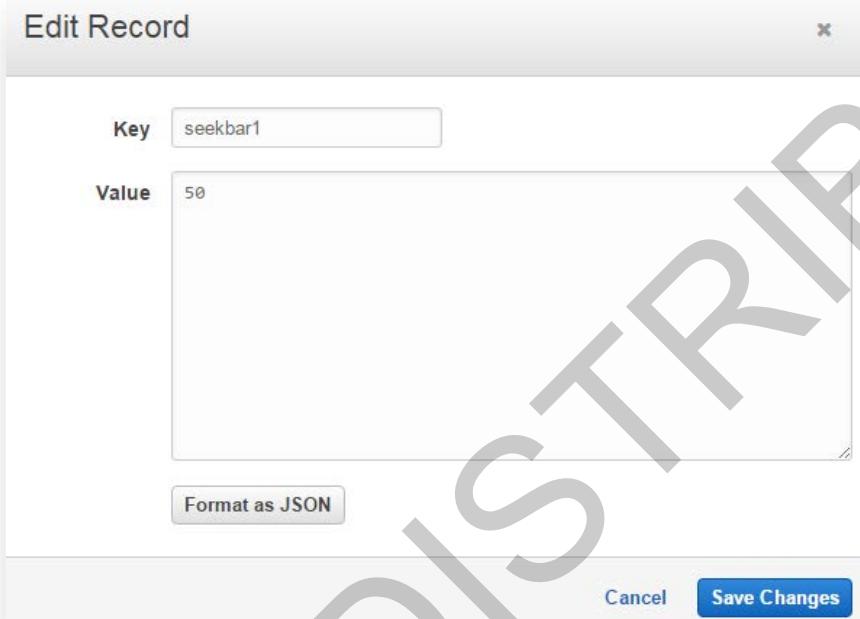
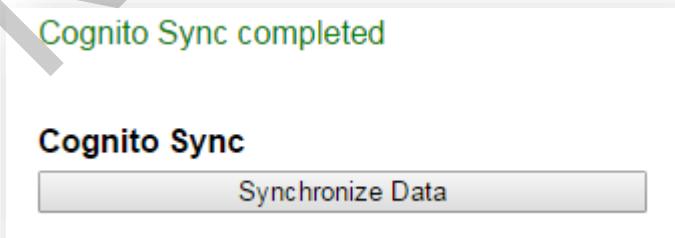


The page will then refresh its gadgets with the Amazon Cognito dataset. Of course, the dataset doesn't exist yet, so the gadgets will be zero.

2.3.6	<p>Slide one of the sliders to a position, and then click Synchronize Data to push the change to Amazon Cognito.</p> 
2.3.7	<p>When the synchronization is complete, copy the Amazon Cognito Identity from the web page, into your clipboard:</p> <p>Cognito Identity: us-east-1:49aed678-6876-4bd5-b588-c0e5085a873d</p> <p>Note that this is not the Amazon Cognito Identity Pool Id, this is the Amazon Cognito Identity of the user logged on via Facebook.</p> <p>Copy the value into your clipboard. So you will have something like this in your clipboard:</p> <pre>us-east-1:36159b32-ab31-48f7-8fc2-498a660a7620</pre> <p>Note that your identity will be different!</p>

2.3.8	Navigate to the Cognito page in your AWS Management Console .
2.3.9	Click on the heading/link ‘Bootcamp2015’
	
2.3.10	The page will refresh. Choose Identity Browser from the links on the left:

2.3.11	<p>Paste in the Amazon Cognito identity from your clipboard and click Search.</p> 												
2.3.12	<p>Click on the dataset named Bootcamp2015_Test</p>  <table border="1"> <thead> <tr> <th></th> <th>Dataset name</th> <th>Date created (UTC)</th> <th>Last modified (UTC)</th> <th>Storage (KB)</th> <th>Records</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>Bootcamp2015Test</td> <td>2015-05-21T08:47:31Z</td> <td>2015-05-21T08:47:31Z</td> <td>0.0</td> <td>2</td> </tr> </tbody> </table>		Dataset name	Date created (UTC)	Last modified (UTC)	Storage (KB)	Records	<input type="checkbox"/>	Bootcamp2015Test	2015-05-21T08:47:31Z	2015-05-21T08:47:31Z	0.0	2
	Dataset name	Date created (UTC)	Last modified (UTC)	Storage (KB)	Records								
<input type="checkbox"/>	Bootcamp2015Test	2015-05-21T08:47:31Z	2015-05-21T08:47:31Z	0.0	2								
2.3.13	<p>Notice that the values you set on the web page have been pushed into the Amazon Cognito Sync Store.</p>  <table border="1"> <thead> <tr> <th></th> <th>Key</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>seekbar1</td> <td>53</td> </tr> <tr> <td><input type="checkbox"/></td> <td>seekbar2</td> <td>67</td> </tr> </tbody> </table>		Key	Value	<input type="checkbox"/>	seekbar1	53	<input type="checkbox"/>	seekbar2	67			
	Key	Value											
<input type="checkbox"/>	seekbar1	53											
<input type="checkbox"/>	seekbar2	67											

2.3.14	<p>Now, make a change to the dataset directly in the Sync Store. For example, click on seekbar1. Change the value to any value other than the current one and click Save Changes</p> 
2.3.15	<p>Note that the change won't be saved until you synchronize. Click the Synchronize button</p>
2.3.16	<p>Now go back to the web page, and click Synchronize Data. When the synchronization completes, the gadgets will update their position based on the values in the dataset.</p> 

DO NOT DISTRIBUTE

Lab 2: Creating an Amazon Kinesis Stream and updating IAM roles

Overview	In this lab you will setup an Amazon Kinesis Stream and use AWS CloudFormation to set up IAM to grant appropriate permissions for Cognito roles to be able to access this stream, as well as for all other resources that we will be using in the Bootcamp. You will use AWS CloudFormation to streamline the process, reduce errors, and save time.
Objectives	After completing this lab, you will be able to: <ul style="list-style-type: none">• Create Kinesis Stream• Create a stack in AWS CloudFormation to update IAM Roles
Pre-requisites	This lab requires: <ul style="list-style-type: none">• Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">◦ Note: The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.• For Microsoft Windows users: Administrator access to the computer.• An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).
Duration	20 minutes

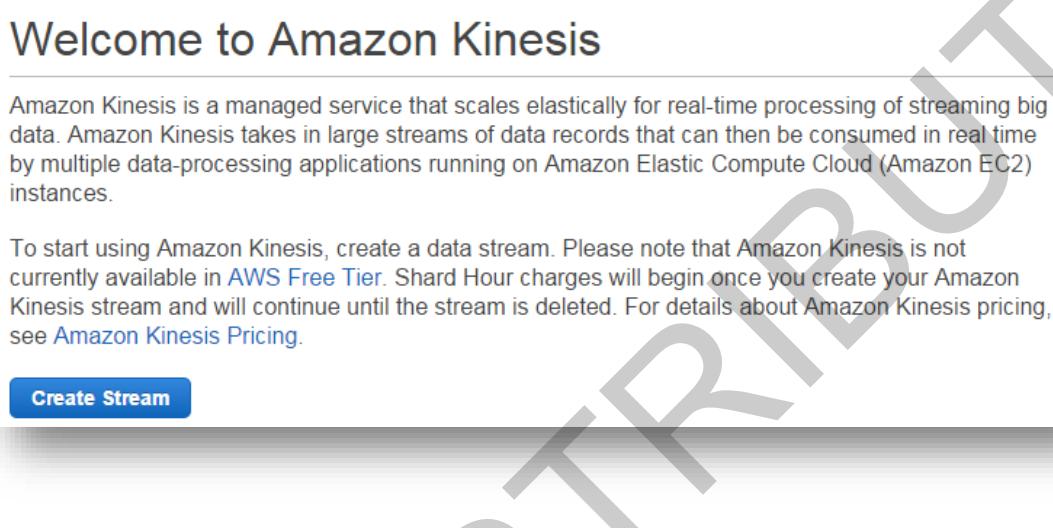
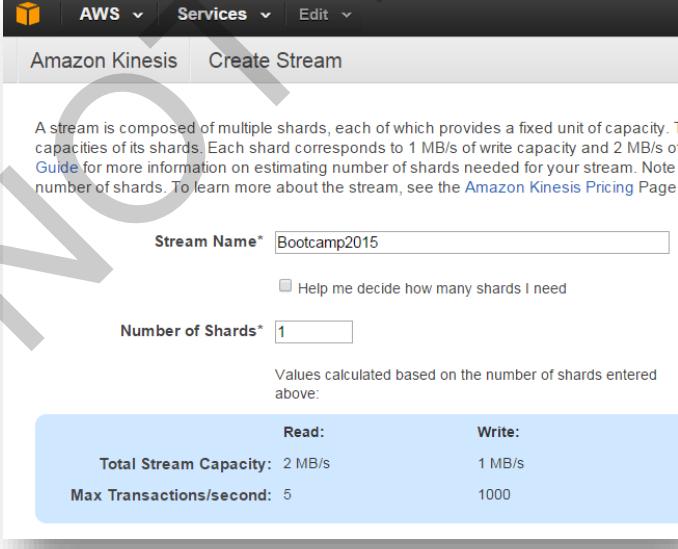
Task 1: Creating an Amazon Kinesis Stream

Overview	In this Bootcamp, we will use Amazon Kinesis to handle high-velocity telemetry in our application. We create an Amazon Kinesis Stream to continuously capture data from our IoT device.
-----------------	---

Task 1-1: Create an Amazon Kinesis Stream

Overview	In this section you will create an Amazon Kinesis Stream to be used for this Bootcamp.
-----------------	--

Step	Instruction
1.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Analytics - Kinesis.</p> <p>All AWS Services</p> <p>Compute</p> <p>Storage & Content Delivery</p> <p>Database</p> <p>Networking</p> <p>Administration & Security</p> <p>Analytics</p> <p>Application Services</p> <p> EMR</p> <p>Amazon Elastic MapReduce lets you perform big data tasks such as web indexing, data mining, and log file analysis.</p> <p> Kinesis</p> <p>Amazon Kinesis is a managed service that scales elastically for real-time processing of streaming big data.</p>

1.1.2	<p>Click the Create Stream button.</p>  <p>The screenshot shows the 'Welcome to Amazon Kinesis' page. It includes a brief introduction about Amazon Kinesis and its capabilities, followed by a note that it's currently not available in the AWS Free Tier. A prominent blue 'Create Stream' button is centered at the bottom of the page.</p>
1.1.3	<p>Enter the stream name as Bootcamp2015</p> <p>and choose the Number of Shards as 1. Then click Create. This will take a few moments to process.</p>  <p>The screenshot shows the 'Create Stream' configuration page. The 'Stream Name*' field is filled with 'Bootcamp2015'. The 'Number of Shards*' field has '1' entered. Below the form, a note states: 'Values calculated based on the number of shards entered above.' At the bottom, the calculated capacities are listed: 'Total Stream Capacity: 2 MB/s' and 'Max Transactions/second: 5' under the 'Read:' column, and '1 MB/s' and '1000' under the 'Write:' column.</p>

Task 2: Creating a new IAM Role for use by AWS Lambda

Overview

In this task, we will create all the policies and permissions required to run our Connected Domestic Appliance Network application, using the ‘least-privilege’ approach whereby we only grant the minimum permissions for access to each resource. We will use AWS IAM to ensure access to our cloud resources is secure.

In order to save time during our Bootcamp, instead of setting up all the IAM roles manually, we will use AWS CloudFormation and an appropriate template to automate the security setup.

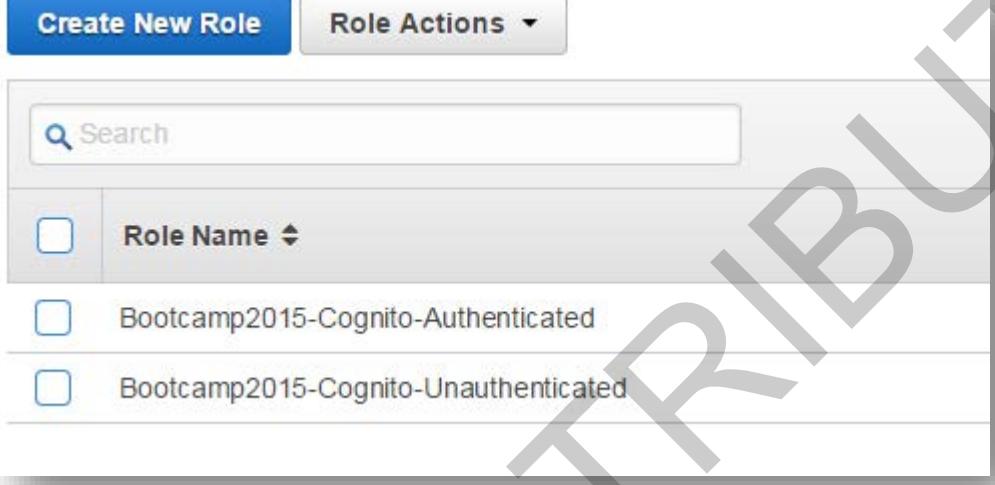
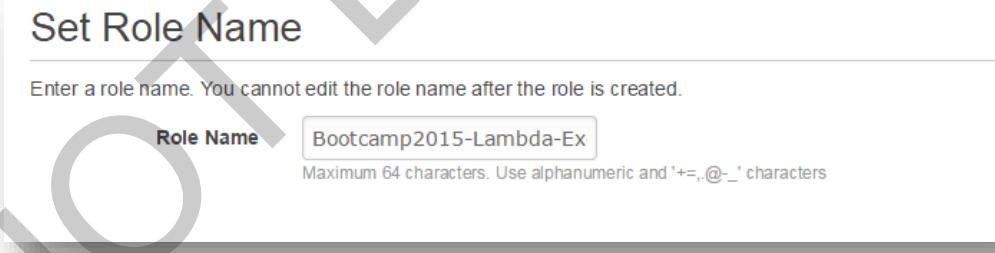
Task 2-1: Create IAM Role for Amazon Lambda functions

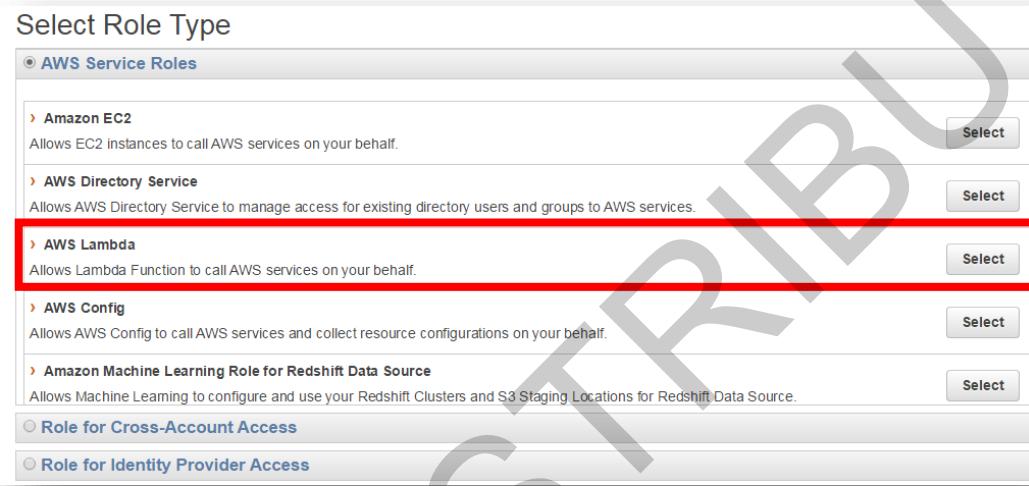
Overview

When the Amazon Lambda functions we use in our Bootcamp execute, they will need the correct IAM permissions to read from an Amazon Kinesis stream, send messages via SNS, read SQS messages, etc. You can grant these permissions by creating an IAM role (also referred to as an execution role) and assigning the appropriate policies and permissions. AWS Lambda assumes this role when executing your Lambda function on your behalf, and inherits the associated policies and permissions.

In this section, you will create an IAM role using a predefined role type and access policy. In the next section, you can use AWS CloudFormation to add the policies to the role

Step	Instruction
2.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Administration & Security - IAM.</p>  <p>The screenshot shows the AWS Management Console navigation bar. The path to IAM is highlighted with a blue arrow: All AWS Services → Administration & Security → IAM.</p> <ul style="list-style-type: none"> All AWS Services Compute Storage & Content Delivery Database Networking Administration & Security Analytics Application Services Deployment & Management Mobile Services Enterprise Applications <p>IAM</p> <p>AWS Identity and Access Management (IAM) lets you securely control access to AWS services and resources.</p> <p>Directory Service</p> <p>AWS Directory Service provides managed directories in the cloud.</p> <p>Trusted Advisor</p> <p>AWS Trusted Advisor inspects your AWS environment and finds opportunities to save money, improve system performance and reliability, or help close security gaps.</p>

2.1.2	<p>Click the Roles button and click Create New Role.</p>  <p>The screenshot shows the 'Role Actions' dropdown menu open, with the 'Create New Role' button highlighted in blue. Below it is a search bar labeled 'Search'. A table lists two roles: 'Bootcamp2015-Cognito-Authenticated' and 'Bootcamp2015-Cognito-Unauthenticated', each with a checkbox next to its name.</p>
2.1.3	<p>Call the role Bootcamp2015-Lambda-Execution-Role</p> <p>Click the Next Step button</p>  <p>The screenshot shows a 'Set Role Name' dialog box. It contains a text input field labeled 'Role Name' with the value 'Bootcamp2015-Lambda-Ex'. Below the input field is a note: 'Maximum 64 characters. Use alphanumeric and '+, @, _' characters'. There is also a note above the input field: 'Enter a role name. You cannot edit the role name after the role is created.'</p>

2.1.4	<p>In the next screen, under AWS Service Roles, click the Select button in the AWS Lambda section. This will create a trust relationship and allow the role to be assumed by AWS Lambda.</p> 
2.1.5	<p>On the next page in the Policy Type filter, enter lambda to search for AWS Lambda-related policies.</p>

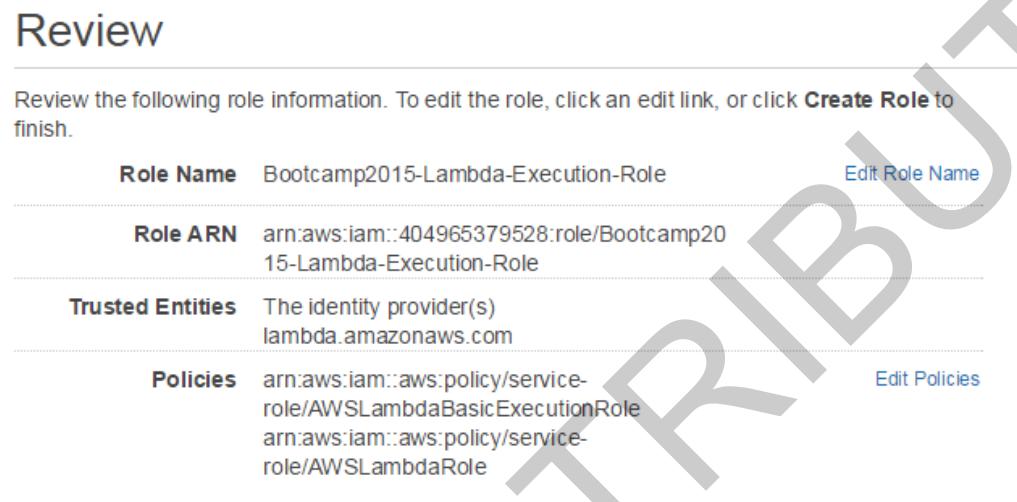
- 2.1.6 Place a tick in the box against the **AWSLambdaBasicExecutionRole** and the **AWSLambdaRole** entries.

Attach Policy

You can have up to two managed policies attached.

	Policy Name	Attached Entities
<input checked="" type="checkbox"/>	AWSLambdaBasicExecutionRole	0
<input type="checkbox"/>	AWSLambdaDynamoDBExecu...	0
<input type="checkbox"/>	AWSLambdaExecute	0
<input type="checkbox"/>	AWSLambdaFullAccess	0
<input type="checkbox"/>	AWSLambdaInvocation-Dynam...	0
<input type="checkbox"/>	AWSLambdaKinesisExecutionR..	0
<input type="checkbox"/>	AWSLambdaReadOnlyAccess	0
<input checked="" type="checkbox"/>	AWSLambdaRole	0

- 2.1.7 Click the **Next Step** button to show the **Review** panel.



- 2.1.8 Click the **Create Role** button, and the role will be created for you.



- 2.1.9 Click the newly created role. You will be redirected back to the Role details panel. You will now see that there is a **Managed Policy** called **AWSLambdaBasicExecutionRole** and **AWSLambdaRole** in the list of Managed Policies:

IAM > Roles > Bootcamp2015-Lambda-Execution-Role

▼ Summary

Role ARN arn:aws:iam::919696375367:role/Bootcamp2015-Lambda-Execution-Role

Instance Profile ARN(s)

Path /

Creation Time 2015-07-12 14:53 UTC+1000

▼ Permissions

Managed Policies

The following managed policies are attached to this role. You can attach up to 2 managed policies.

Attach Policy

Policy Name	Actions
 AWSLambdaBasicExecutionRole	Show Policy
 AWSLambdaRole	Show Policy

Use the 'Show Policy' links to review the permissions that each policy provides, and familiarize yourself with the basic requirements for AWS Lambda to execute in your account.

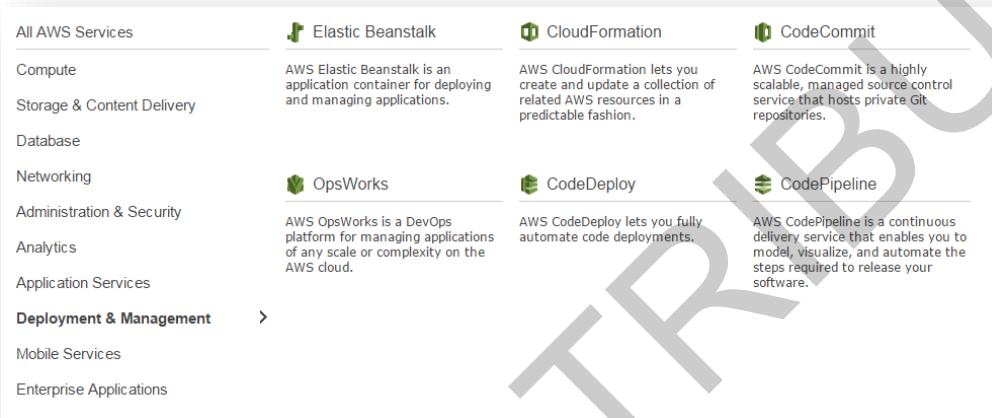
Task 3: Updating IAM Roles for AWS Lambda and Amazon Cognito using an AWS CloudFormation Stack

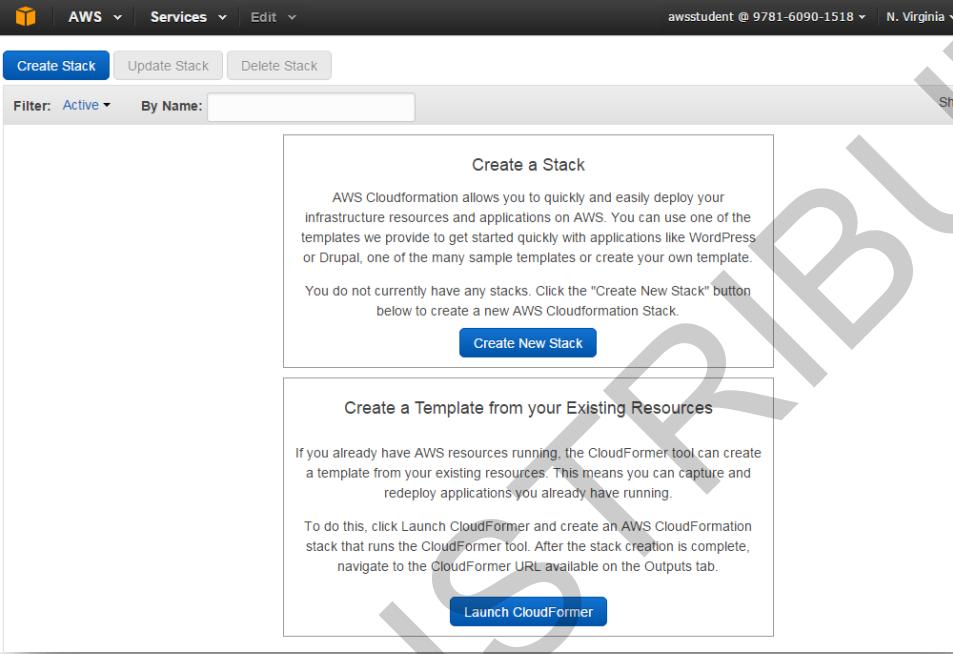
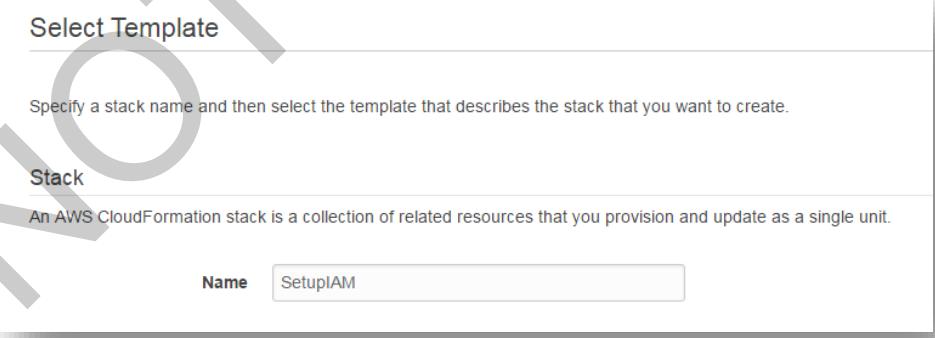
Overview	<p>We will be writing data to the Amazon Kinesis stream from the Unauthenticated IAM role we have assigned to the Amazon Cognito Identity Pool, as data is sent in from the IoT devices. So we need to give the UnAuthenticated role permission to write to the stream.</p> <p>In order to streamline the updating of the IAM roles we use in the Bootcamp, we will add all the relevant permissions we need to the IAM roles in this task. We will do this using the AWS automation tool, AWS CloudFormation.</p>
-----------------	--

Task 3-1: Create AWS CloudFormation Stack

Step	Instruction
------	-------------

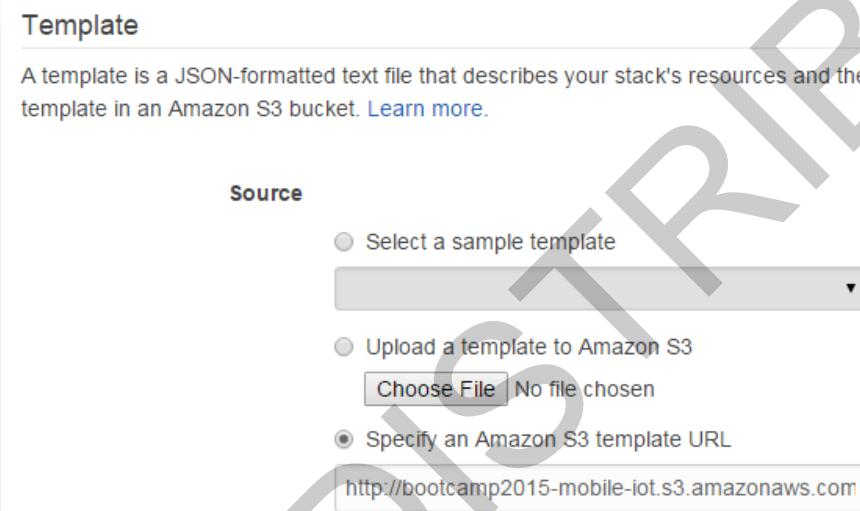
3.1.1 Navigate to the AWS CloudFormation dashboard under **Deployment and Management-> CloudFormation**.



3.1.2	<p>The getting started panel is displayed.</p>  <p>Click Create New Stack.</p>
3.1.3	 <p>Give the stack the name SetupIAM</p>

- 3.1.4 In the Template section, select the **Specify an Amazon S3 template URL** and paste in this URL:

<http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/Desktop/Bootcamp2015-CognitoAuthenticatedPolicies-CloudFormation.template>



Click the **Next** button.

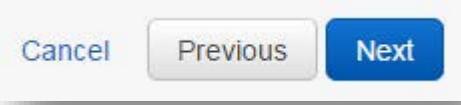
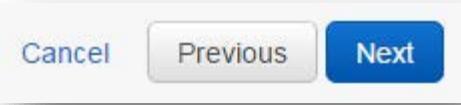
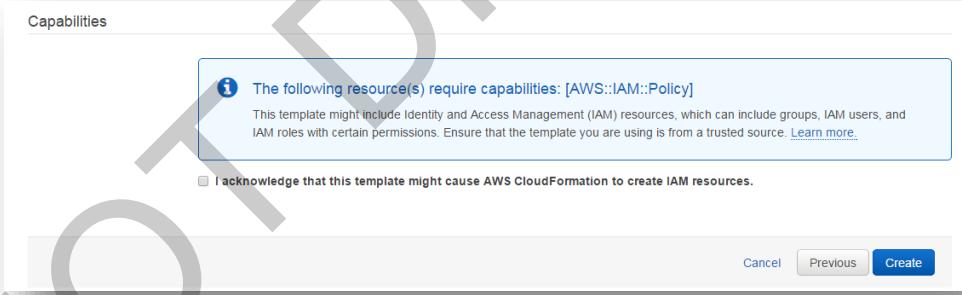
- 3.1.5 In the Specify Parameters screen, leave all the default parameters as they are.

Specify Parameters

Specify values or use the default values for the parameters that are associated with this template.

Parameters

CognitoAuthenticatedRole	Bootcamp2015-Cognito-Authenticated
CognitoUnAuthenticatedRole	Bootcamp2015-Cognito-UnAuthenticated
DynamoDBTableName	Bootcamp2015-BLEDetections
LambdaExecutionRoleName	Bootcamp2015-Lambda-Execution-Role
SnsTopicFromEdison	Bootcamp2015-From-Edison
SNSTopicToMobileApp	Bootcamp2015-To-MobileApp
SqsQueueToEdison	Bootcamp2015-Edison-Command The name of the SQS queue set in SQS_URL_QUEUE_

3.1.6	<p>Click the Next button to continue.</p> 
3.1.7	<p>In the Options Tags screen, click Next.</p> 
3.1.8	<p>In the Review screen, scroll to the bottom, and in the Capabilities section, ensure the 'I acknowledge that this template might cause AWS CloudFormation to create IAM resources' is checked.</p> 

3.1.9	<p>Click Create.</p> <p>AWS CloudFormation will now add the relevant policies to the Role associated with the Amazon Cognito Identity Pool for Authenticated access.</p> <p><i>Note: If you see this error, it means you have forgotten to tick the IAM acknowledgement in the previous step!</i></p> <div data-bbox="388 608 1357 792" style="border: 1px solid red; padding: 10px;"> Error Stack creation error: Requires capabilities : [CAPABILITY_IAM]</div>
-------	--

- 3.1.10 AWS CloudFormation will now commence creating all the IAM policies and attaching them to the roles as required in the Bootcamp.
- You can click on the 'refresh' icon at the top right of the stack list, to refresh the view, or wait for the interface to update as it does periodically.

Stack Name				Created Time	Status	Description
<input checked="" type="checkbox"/>	SetupIAM			2015-08-06 09:26:40 UTC+1000	CREATE_IN_PROGRESS	

This will take a minute to complete, and then you will see a 'Complete' message.

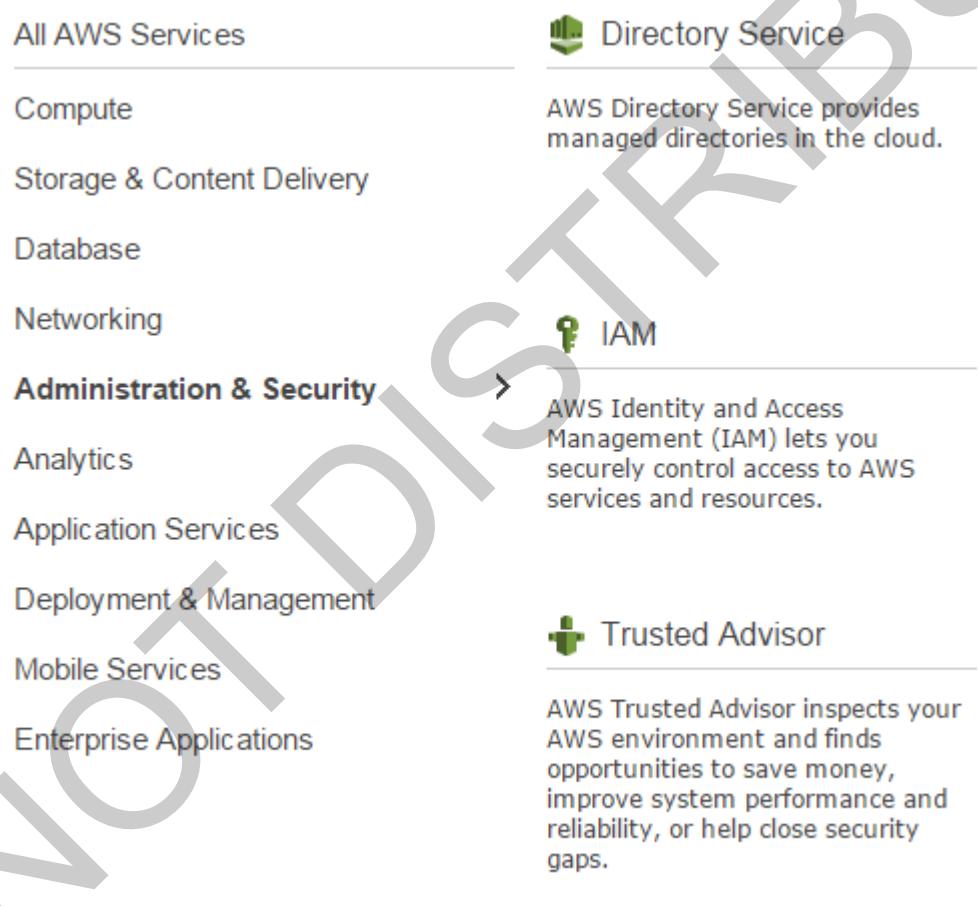
Stack Name			Created Time	Status
<input checked="" type="checkbox"/>	SetupIAM		2015-08-06 09:37:46 UTC+1000	CREATE_COMPLETE

3.1.11

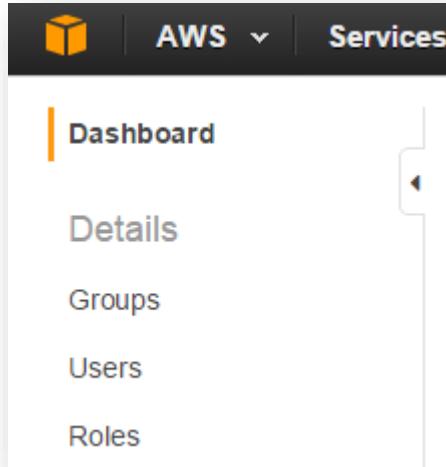
Optional Task (If Time Permits):

You can view the changes the AWS CloudFormation template has made to your AWS account, by browsing to the IAM manager in the AWS console.

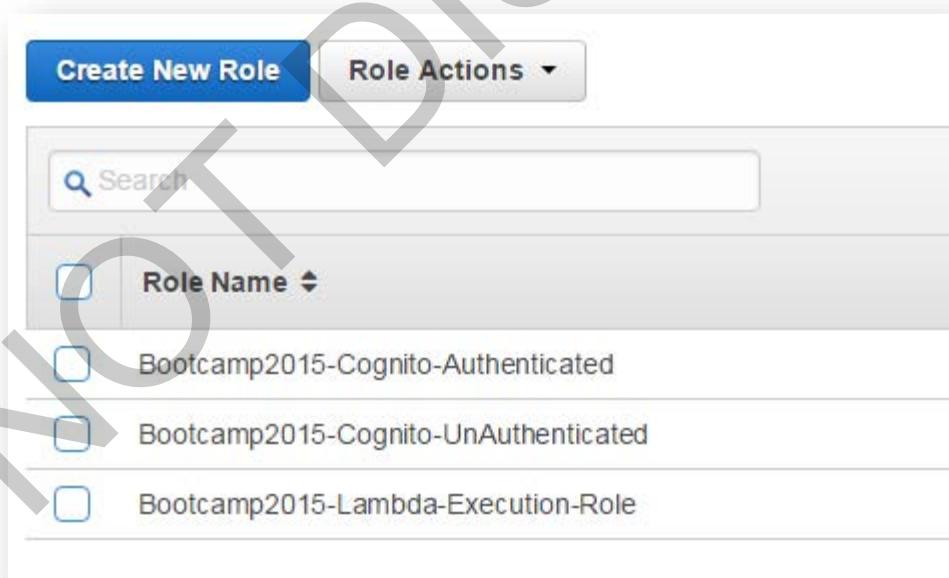
Login to the AWS Management Console from your Qwiklabs Environment and click the **Services -> Administration & Security -> IAM** button



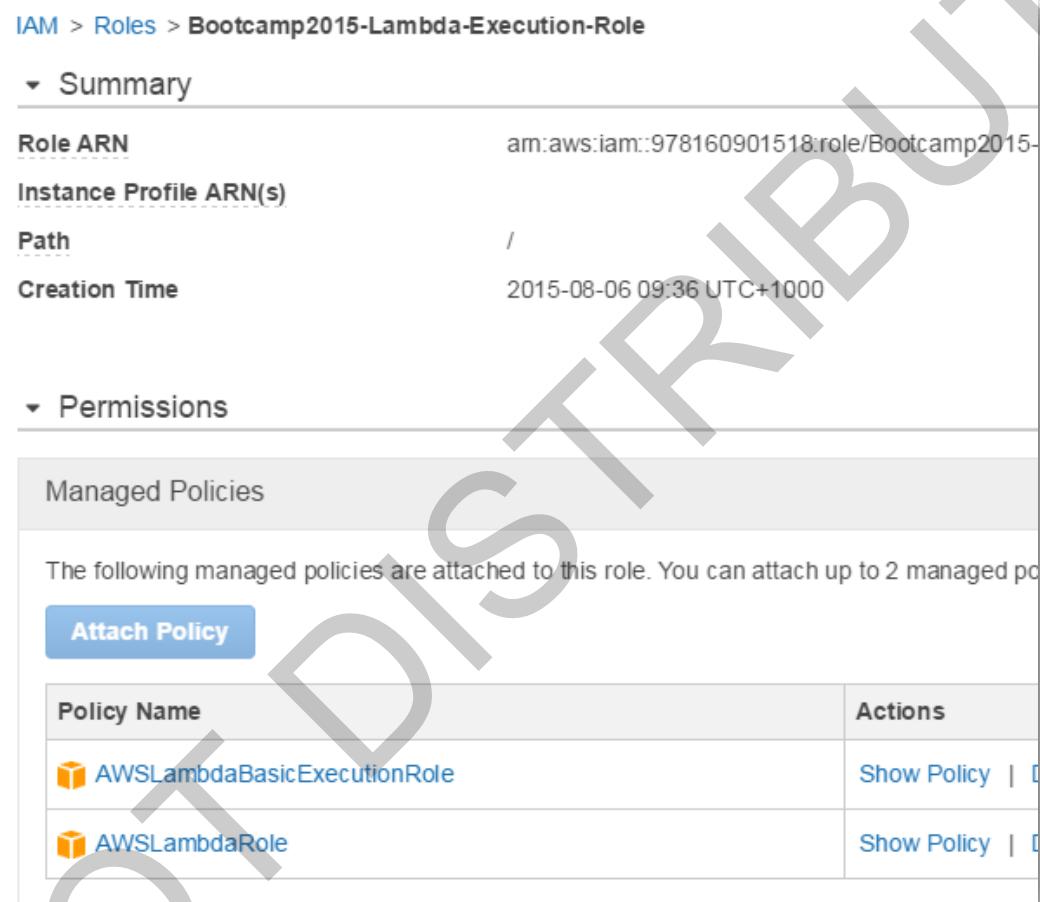
3.1.12 In the left navigation panel, click **Roles**.



3.1.13 The panel will refresh.



Click the last entry, the **Bootcamp2015-Lambda-Execution-Role**

3.1.14	<p>The panel will again refresh. This time it will show the permissions and policies for the IAM role.</p>  <p>IAM > Roles > Bootcamp2015-Lambda-Execution-Role</p> <p>Summary</p> <table border="1"> <tbody> <tr> <td>Role ARN</td> <td>arn:aws:iam::978160901518:role/Bootcamp2015-</td> </tr> <tr> <td>Instance Profile ARN(s)</td> <td></td> </tr> <tr> <td>Path</td> <td>/</td> </tr> <tr> <td>Creation Time</td> <td>2015-08-06 09:36 UTC+1000</td> </tr> </tbody> </table> <p>Permissions</p> <p>Managed Policies</p> <p>The following managed policies are attached to this role. You can attach up to 2 managed policies.</p> <p>Attach Policy</p> <table border="1"> <thead> <tr> <th>Policy Name</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>AWSLambdaBasicExecutionRole</td> <td>Show Policy Delete</td> </tr> <tr> <td>AWSLambdaRole</td> <td>Show Policy Delete</td> </tr> </tbody> </table>	Role ARN	arn:aws:iam::978160901518:role/Bootcamp2015-	Instance Profile ARN(s)		Path	/	Creation Time	2015-08-06 09:36 UTC+1000	Policy Name	Actions	AWSLambdaBasicExecutionRole	Show Policy Delete	AWSLambdaRole	Show Policy Delete
Role ARN	arn:aws:iam::978160901518:role/Bootcamp2015-														
Instance Profile ARN(s)															
Path	/														
Creation Time	2015-08-06 09:36 UTC+1000														
Policy Name	Actions														
AWSLambdaBasicExecutionRole	Show Policy Delete														
AWSLambdaRole	Show Policy Delete														
3.1.15	<p>Under the Permissions section, you will see the Managed Policies you selected when you created the role in the early steps.</p>														

- 3.1.16 If you scroll lower on the page, you will see the Inline Policies the AWS CloudFormation template automatically applied to the role:

Inline Policies

This view shows all inline policies that are embedded in this role.

Create Role Policy

Policy Name
Bootcamp2015-GetOpenIdTokenForDeveloperIdentity
Bootcamp2015-Lambda-Access-Kinesis
Bootcamp2015-Lambda-DynamoDB
Bootcamp2015-Lambda-To-Mobile

You can click on the Edit Policy link against each to view the actual policies

[Show Policy](#) | [Edit Policy](#) |

For example:

Policy Name

Bootcamp2015-Lambda-Access-Kinesis

Policy Document

```
1 {  
2     "Statement": [  
3         {  
4             "Resource": "arn:aws:kinesis:us-east-1:  
5             "Action": [  
6                 "kinesis:DescribeStream",  
7                 "kinesis>ListStreams",  
8                 "kinesis:GetShardIterator",  
9                 "kinesis:GetRecords"  
10            ],  
11            "Effect": "Allow"  
12        }  
13    ],  
14    "Version": "2012-10-17"  
15}
```

Lab 3: Configuring Your Edison

Overview	In this lab you will configure your Intel Edison device and prepare it for the Bootcamp.
Objectives	After completing this lab, you will be able to: <ul style="list-style-type: none">• Connect your Edison to the WiFi• Name your Edison• Set the root password• Test the Edison Bluetooth hardware• Configure your Edison for use in the Bootcamp
Pre-requisites	This lab requires: <ul style="list-style-type: none">• Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">◦ Note The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.• For Microsoft Windows users: Administrator access to the computer.• An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).• An SSH client, such as PuTTY.• 2 available USB ports on your notebook computer.
Duration	30 minutes

Task 1: Setting up your Edison Mini Breakout Board

Overview

In this guide, you'll connect the Intel® Edison module to an [Intel® Edison mini breakout board](#). Your Edison Package contents should look like the picture below. Note that the breakout board has two micro USB ports; The top port is used to create a terminal connection by serial over USB only. The bottom port is for power and USB communication.



Task 1-1: Assemble Your Board

Overview

At the end of this section, you should have an assembled Intel Edison board. You can skip this setup if your Edison is already assembled.

Step

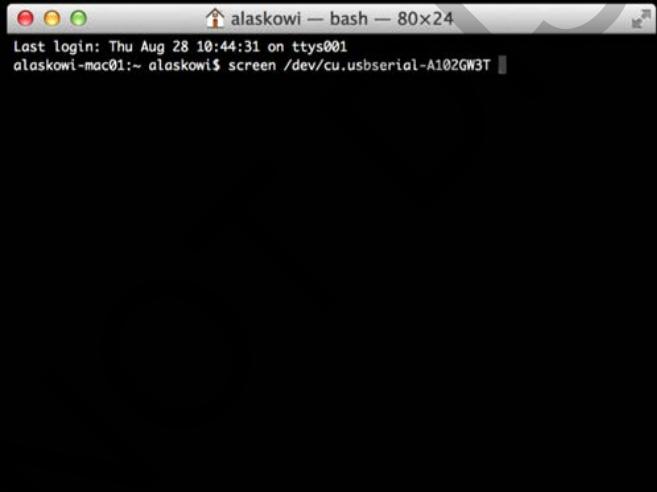
Instruction

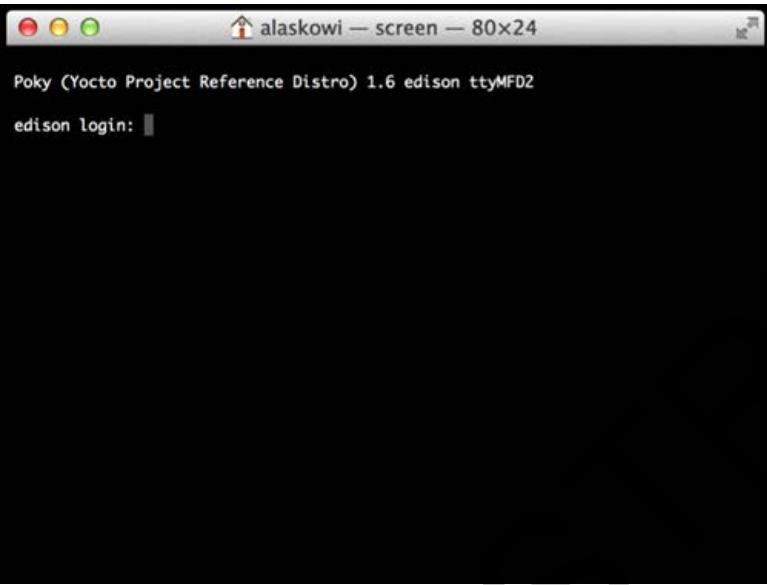
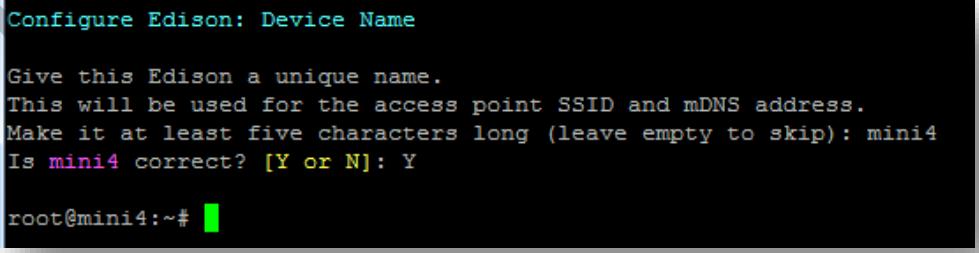
1.1.1	<p>Place the Intel Edison module on the breakout board, lining up the holes on the module with the screws on the breakout board. Press down on the module at the upper left corner and just below the words "What will you make?" until you feel it click into place.</p> <p><i>Treat the board gently!</i></p> 
1.1.2	<p>Use the two hex nuts to secure the module to the expansion board. Hand-tighten the hex nuts onto the two screws that protrude through the module.</p> 

1.1.3	<p>Orientate your board so the SoC (System on a Chip) is facing up, and the USB connectors are on the right as you face it.</p> <p>Plug in one of the micro-USB cables to the bottom USB connector on the expansion board. Plug in the other end of the USB cable to your computer. A green light should light up on the expansion board. If it doesn't, check your connection.</p>
1.1.4	<p>Wait a moment for the board to boot up. You will know that the board is fully initialized when your computer mounts a new drive (much like inserting a SD card into your computer).</p> 
1.1.5	<p>Plug in your second USB cable to the top USB connector on the board.</p> <p>Note: If you do not see a new drive, it is likely that the board isn't getting enough power from the USB port. Plug in your laptop's AC adapter (if you are connecting the board to a laptop), try a different USB port on your computer, or try using a USB hub that has its own power supply.</p> 

Task 1-2: Mac OS X – Configure the Edison

Overview	<p>In this task, you will connect to your Edison using the Serial port, and then set the root user password</p> <p>This task is specific to Mac OSX. For instruction on Windows, see the next Task.</p>
-----------------	---

Step	Instruction
1.2.1	Connect your board as shown in 1.1.4 and 1.1.5
1.2.2	Open a new Terminal window. Type <code>screen /dev/cu.usbserial</code> , then press Tab .
	
1.2.3	Type <code>115200 -L</code> . Press Enter

1.2.4	<p>A black screen displays. Press Enter until you see the 'edison login:' prompt:</p> 
1.2.5	<p>Type root and press Enter. You are now logged in to the device as the root user</p>
1.2.6	<p>Assign your board a name by entering the command: <pre>configure_edison --name</pre> <p>Note that there are two dashes before name (--name)</p> <p>Follow the prompts on screen. Give your device a name like AdamsMiniEdison or whatever makes sense to you</p>  </p>

1.2.7

Create a login password for your board by entering the command:

```
configure_edison --password
```

Note that there are two dashes before password (--password)

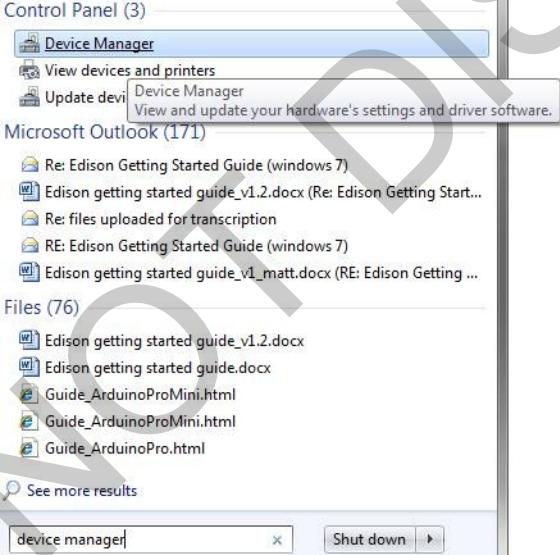
Follow the prompts on screen to set the root password. We recommend setting the password to:

```
edison2015
```

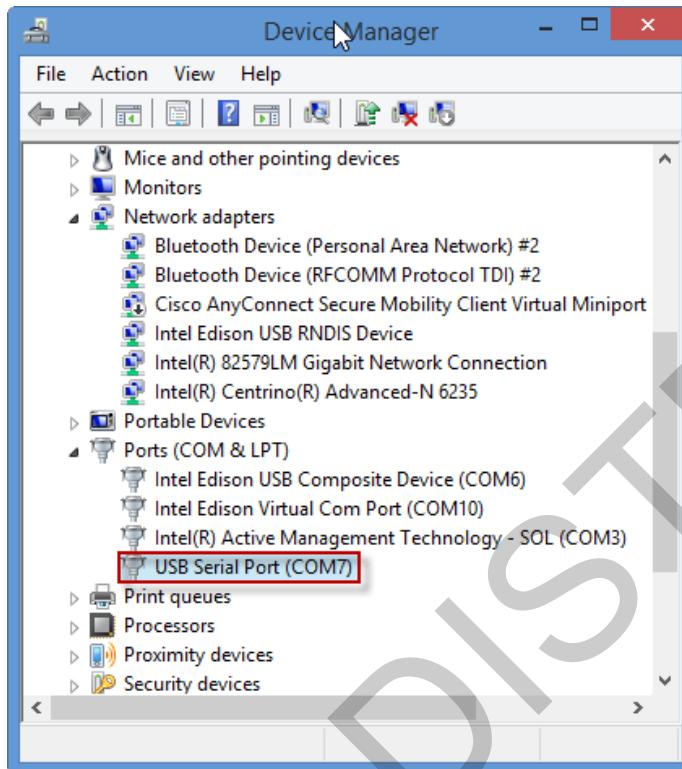
By setting the password exactly as shown above, it will make it easier for a proctor to assist you during the Bootcamp, should you need assistance.

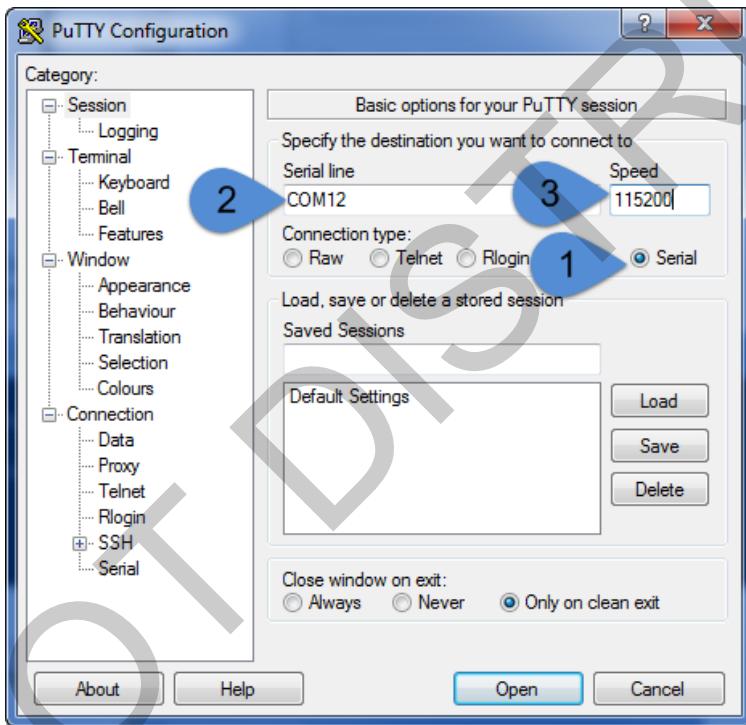
Task 1-3: Windows – Configure the Edison

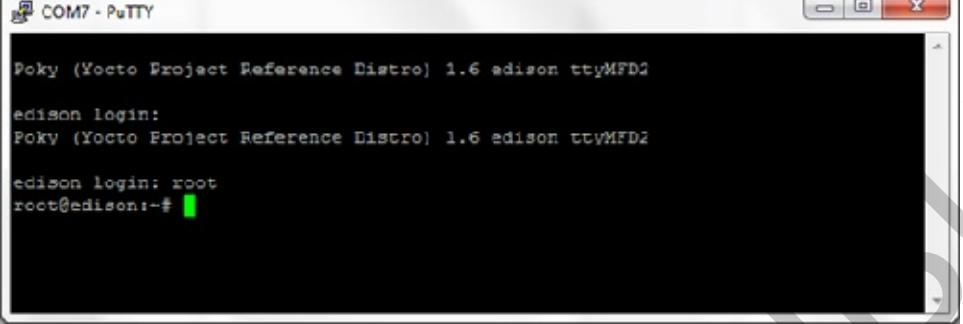
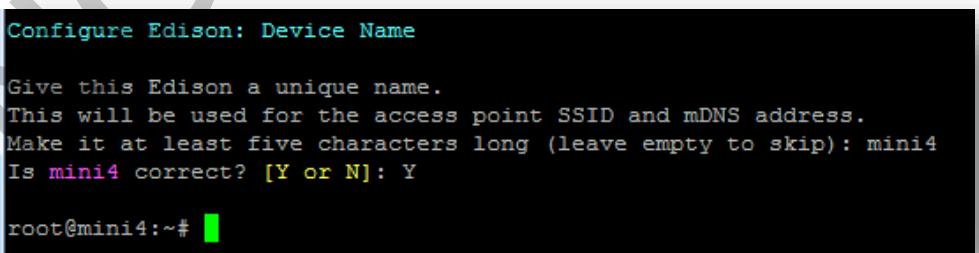
Overview	<p>In this task, you will connect to your Edison using the Serial port, and then set the root user password</p> <p>These instructions are specific to Windows users – see the previous Task to learn how to do this for Mac OSX</p>
-----------------	---

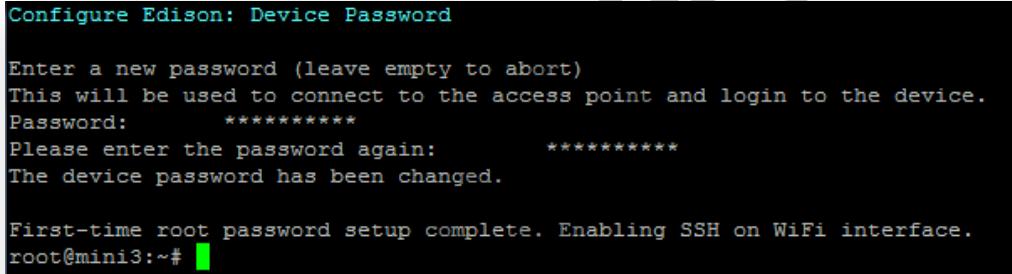
Step	Instruction
1.3.1	Connect your board as shown in 1.1.4 and 1.1.5
1.3.2	<p>Open Device Manager by clicking Start and typing Device Manager.</p>  A screenshot of the Windows Control Panel search interface. The search bar at the bottom contains the text "device manager". Above the search bar, there is a list of results. The first result, "Device Manager", is highlighted with a blue selection bar. Other results include "View devices and printers", "Update devi", "Microsoft Outlook (171)", and "Files (76)". Below the search bar, there are "Shut down" and "▶" buttons.

- 1.3.3 Scroll down to **Ports (COM & LPT)** and take note of the **COM** number listed under **USB Serial Port (COM#)**. In this example, it is **COM7**.



1.3.4	<p>Open the PuTTY.exe application that you downloaded and installed as part of the prerequisites for the Bootcamp. If you haven't done this, please refer to the prerequisites documents for assistance</p> <p>Once PuTTY is launched:</p> <ol style="list-style-type: none"> Under Connection type, select Serial. In the Serial line field, enter the COM# for your board, such as COM12 in this example. In the Speed field, type 115200. 
1.3.5	<p>Click Open. A blank screen is displayed. Press Enter as many times as you need to get the device's attention, so it shows the 'edison login:' prompt</p>

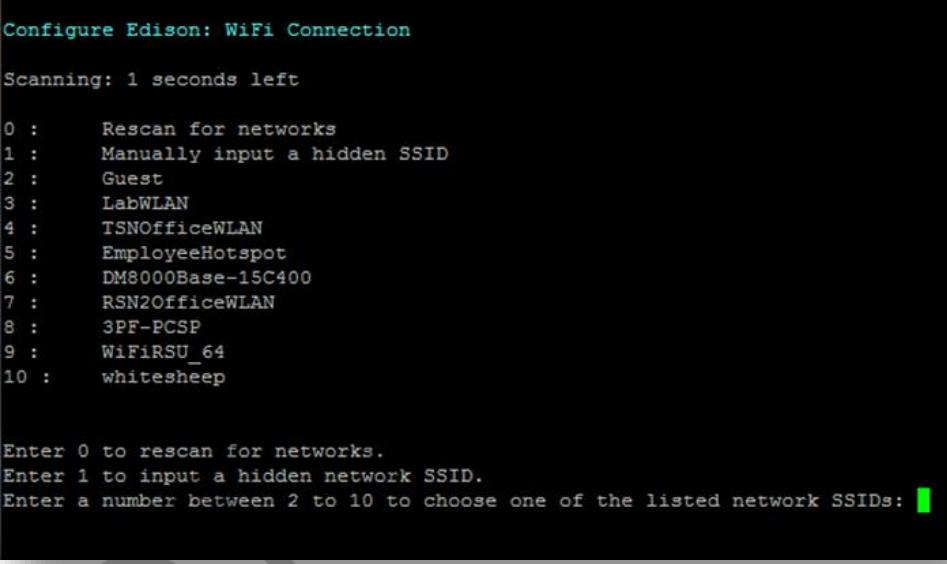
1.3.6	<p>Type <code>root</code> and press Enter</p>  <pre data-bbox="339 418 979 593"> Poky (Yocto Project Reference Distro) 1.6 edison ttyMFD2 edison login: Poky (Yocto Project Reference Distro) 1.6 edison ttyMFD2 edison login: root root@edison:~# </pre>
1.3.7	<p>Assign your board a name by entering the command:</p> <pre data-bbox="323 1142 698 1174">configure_edison --name</pre> <p>Note that there are two dashes before name (<code>--name</code>)</p> <p>Follow the prompts on screen. Give your device a name like AdamsMiniEdison or whatever makes sense to you</p>  <pre data-bbox="355 1396 1334 1649"> Configure Edison: Device Name Give this Edison a unique name. This will be used for the access point SSID and mDNS address. Make it at least five characters long (leave empty to skip): mini4 Is <code>mini4</code> correct? [Y or N]: Y root@mini4:~# </pre>

<p>1.3.8 Create a login password for your board by entering the command:</p> <pre>configure_edison --password</pre> <p>Note that there are two dashes before password (--password)</p> <p>Follow the prompts on screen to set the root password. We recommend setting the password to:</p> <pre>edison2015</pre> <p>By setting the password exactly as shown above, it will make it easier for a proctor to assist you during the Bootcamp, should you need assistance.</p>  <pre>Configure Edison: Device Password Enter a new password (leave empty to abort) This will be used to connect to the access point and login to the device. Password: ***** Please enter the password again: ***** The device password has been changed. First-time root password setup complete. Enabling SSH on WiFi interface. root@mini3:~#</pre>

Task 1-4: Windows and OSX: Connect Your Edison to Wi-Fi

Overview	This guide contains steps to set up network access to your Intel® Edison board and to obtain an IP address.
-----------------	---

Step	Instruction
1.4.1	<p>Establish a serial communication session with your board (refer to previous sections for your relevant platform) and log on as root.</p> <p><i>If you are following on from the previous tasks, you will already have a serial connection, and be logged on as root.</i></p>

1.4.2	<p>To configure your Wi-Fi, enter the command:</p> <pre>configure_edison --wifi</pre> <p>Note that there are two dashes before wifi (--wifi)</p>
1.4.3	<p>When asked if you want to set up WiFi, type Y and press Enter.</p>
1.4.4	<p>Your board will scan for Wi-Fi networks for 10 seconds. When it is finished, a list of available networks will be displayed. If you don't see any networks, enter 0 to rescan.</p>  <pre>Configure Edison: WiFi Connection Scanning: 1 seconds left 0 : Rescan for networks 1 : Manually input a hidden SSID 2 : Guest 3 : LabWLAN 4 : TSNOfficeWLAN 5 : EmployeeHotspot 6 : DM8000Base-15C400 7 : RSN2OfficeWLAN 8 : 3PF-PCSP 9 : WiFiRSU_64 10 : whitesheep Enter 0 to rescan for networks. Enter 1 to input a hidden network SSID. Enter a number between 2 to 10 to choose one of the listed network SSIDs: [cursor]</pre>

1.4.5	<p>Choose the network you would like to connect to, type the corresponding number from the list, and press Enter. To confirm your entry, type Y and press Enter. In this example, to connect to the kafka network, enter 16.</p> <pre>13 : STSPDX 14 : belkin.19a 15 : WiFIRSU_953c2 16 : kafka</pre> <p>Enter 0 to rescan for networks. Enter 1 to input a hidden network SSID. Enter a number between 2 to 16 to choose one of the listed networks.</p> <p>Your instructor will tell you which of the available networks you should connect to. Be sure to only connect to the network you are instructed to connect to.</p>
1.4.6	<p>If your network requires a password or other information, enter the appropriate network credentials.</p> <p>Your instructor will tell you the correct SSID and passphrase for the network.</p>
1.4.7	<p>The board will attempt to make a connection to the network. When you see a Done message, your board is connected to a Wi-Fi network.</p> <pre>Enter a number between 2 to 16 to choose one of the listed network SSIDs: 16 Is kafka correct? [Y or N]: y Password must be between 8 and 63 characters. What is the network password?: ***** Initiating connection to kafka. Please wait... Attempting to enable network access... check 'wpa_cli status' after a minute to confirm. Done. Please connect your computer to the same network as this device and go to http://10.0.0.26 root@myedison:~#</pre>  <p>IP Address</p>

- 1.4.8 Note the IP Address, as shown in the image above. This is your board's IP Address. Alternately, enter the command:

```
ifconfig
```

```
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

wlan0      Link encap:Ethernet HWaddr fc:c2:de:3e:91:b4
           inet addr:192.168.0.105 Bcast:0.0.0.0 Mask:255.255.255.0
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
             RX packets:9 errors:0 dropped:0 overruns:0 frame:0
             TX packets:45 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:2664 (2.6 KiB) TX bytes:11045 (10.7 KiB)
```

Make note of your **wlan0** IP address, as shown above.

Note: In a later step, we will add a service to the Edison that will cause the device to 'register' with our Central Web Site and publish the Edison's allocated IP address. This will make it easier to find the IP address of your Edison later, should the DHCP server on your network allocate a different address

- 1.4.9 To verify connectivity, enter the following and you should see a file download:

```
wget aws.amazon.com
```

```
root@mini4:~# wget www.amazon.com
--2015-09-12 03:26:35-- http://www.amazon.com/
Resolving www.amazon.com... 176.32.103.205
Connecting to www.amazon.com|176.32.103.205|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html.1'

[          <=>                               ] 376,916      151KB/s   in 2.4s

2015-09-12 03:26:39 (151 KB/s) - 'index.html.1' saved [376916]

root@mini4:~#
```

1.4.10

If you are having problems connecting, try running the following commands in a serial communication session with your board:

```
ifconfig usb0 down  
ifconfig wlan0 down  
ifconfig usb0 up  
ifconfig wlan0 up
```

Task 2: Configuring Edison for Bootcamp

Overview

In this section you will download the Bootcamp files, register your device with the Central Bootcamp Website and connect to your Edison via the WiFi network.

After you have downloaded the Edison.tar bundle as described below, the various other tasks move the contents of the bundle into the relevant positions on the Edison's filesystem. For example, you will move the lab implementation files into the /home/root folder, the Bootcamp-config.js configuration file in to /etc and configure a service to startup on boot – as well as other filesystem configuration. You will also run the Node Package Manager 'npm' utility to install the various dependencies needed to run the NodeJS implementation files.

Take care to read the various instructions closely. While some steps look like repeats of previous steps (especially in the case of running the npm utility) they are no repeats. Be sure to execute every step as prescribed.

If you get stuck, please refer to a Proctor. We have included an automatic configuration script that will help if you get really stuck, so just raise your concern with a Proctor and they can help you run the configuration script and get you up and running.

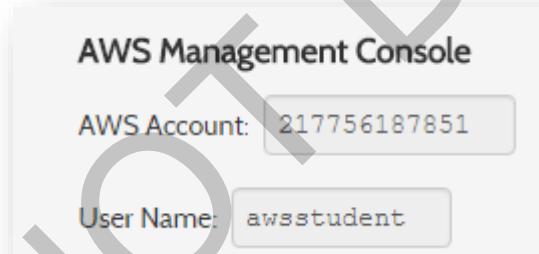
Task 2-1: Download Files and Configure Edison for Bootcamp

Overview

We will download the bootcamp files and run some scripts and configuration to prepare your Edison for this bootcamp.

Step	Instruction
2.1.1	<p>Following on from previous steps, you will already be connected to your Edison via the Serial connection, and be logged on as root</p> <p>On your device terminal enter the following:</p> <pre>cd /home/root wget http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/Edison.tar</pre> <div style="background-color: black; color: white; padding: 10px;"> <pre>root@mini3:~# wget http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/Edison .tar --2015-07-18 03:51:34-- http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/ Edison.tar Resolving bootcamp2015-mobile-iot.s3.amazonaws.com... 54.231.19.9 Connecting to bootcamp2015-mobile-iot.s3.amazonaws.com 54.231.19.9 :80... connected. HTTP request sent, awaiting response... 200 OK Length: 131584 (128K) [application/x-tar] Saving to: 'Edison.tar' 100%[=====] 131,584 139KB/s in 0.9s 2015-07-18 03:51:36 (139 KB/s) - 'Edison.tar' saved [131584/131584] root@mini3:~#</pre> </div>
2.1.2	<p>Untar the file using the following command and move the startup folder to root</p> <pre>tar -xvf Edison.tar mv /home/root/home-root/startup /home/root</pre>
2.1.3	<p>Move the Bootcamp configuration file to its proper place on your Edison.</p> <p>to</p> <pre>/home/root/etc/bootcamp2015-config.js</pre> <p>to</p> <pre>/etc/bootcamp2015-config.js</pre> <p>with this command (which will also update your profile.d file as well)</p> <pre>mv /home/root/etc/* /etc/.</pre>

2.1.4	<p>Before we move on, we will now edit the bootcamp-config.js file to update it with the various values from our account and some decisions you need to make.</p> <p>Open the /etc/bootcamp-config.js file in a text editor on the Edison, such as vi, using the command:</p> <pre>vi /etc/bootcamp2015-config.js</pre> <pre>root@mini3:~# vi /etc/bootcamp2015-config.js</pre>
2.1.5	<p>In the file, you will see several configuration values:</p> <pre>///////// // Configure these values // ///////// var DEVICE_NAME = "Mini Edison"; var AWS_ACCOUNT_ID = "0000000000"; var COGNITO_IDENTITY_POOL_ID = "us-east-1:XXXXXXXXXXXX"; var MOBILE_ANALYTICS_APP_ID = "0000000000"; var API_GATEWAY_ENDPOINT_URL = "https://REPLACE_THIS_WITH_API_GATEWAY_URL_FROM_LAB_GUIDE"; var APPLIANCE_TYPE = 5; var BLE_UUID = "badf00d5dffbb48d2b060d0f5a71096e0"; var GOOGLE_PROJECT_NUMBER = "0000000000"; var MOBILE_PUSH_PROVIDER = "GCM"; // Set to GCM for Android or APNS for iOS mobile push</pre> <p>These values in the supplied default configuration file are set to ‘dummy’ values. You are not able to change all of these dummy values right now, because we haven’t set up all the AWS services we will need yet.</p> <p>We will change the configuration file incrementally as we progress through the Bootcamp. However, in the next steps, we will change the values we have already configured in AWS or base them on decisions you can make now.</p> <p>Use the arrow keys to position over the value you need to change, use the x key to delete characters and the i key to enter insert edit mode.</p>

2.1.6	<pre><code>var DEVICE_NAME = "Mini Edison";</code></pre> <p>DEVICE_NAME</p> <p>Set this to a name that describes you or your Edison. For example, <i>AdamsEdison</i> or <i>AdamLarterDevice</i></p> <p>The name is completely arbitrary, but you should use a name that will be recognizable in a list of device names, and also unique to you.</p>
2.1.7	<pre><code>var AWS_ACCOUNT_ID = "0000000000";</code></pre> <p>AWS_ACCOUNT_ID</p> <p>Change this value to the AWS Account Id for your Qwiklabs account. You can get the Account Id by viewing the Qwiklabs dashboard, where you started the Qwiklab from. An example of the Account Id is shown below:</p>  <p>Note: Your account Id will be different! You cannot use the account Id shown above, you must use the account Id for the Qwiklabs account you are using for the Bootcamp!</p>

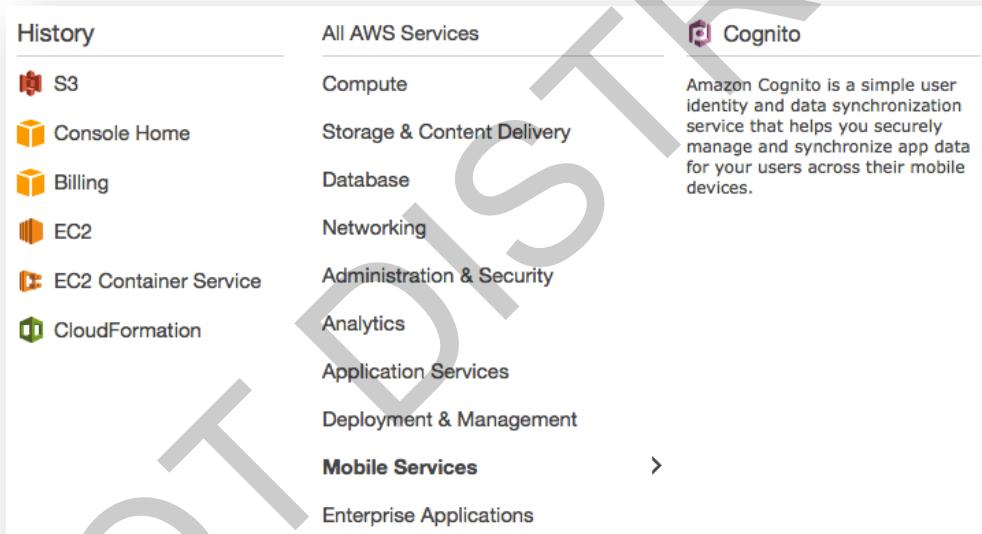
2.1.8

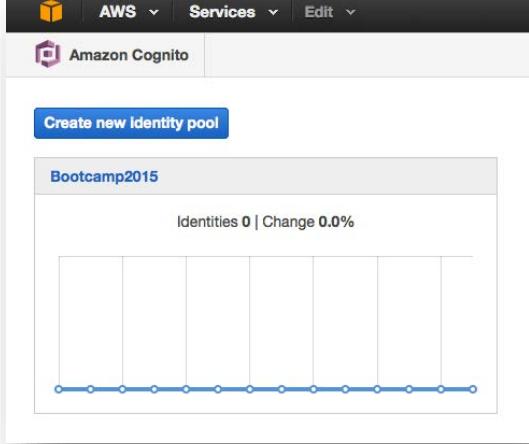
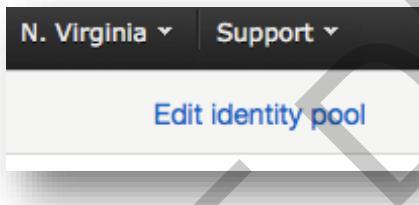
```
var COGNITO_IDENTITY_POOL_ID = "us-east-1:XXXXXXXXXXXX";
```

COGNITO_IDENTITY_POOL_ID

You created a new Cognito Identity Pool in the previous Lab. If you didn't keep the Identity Pool Id handy, you can retrieve it now.

Login to the AWS Management Console from your Qwiklabs Environment and click the **Services - Mobile Services – Cognito**.



2.1.9	<p>Click the Cognito Pool name in this case Bootcamp2015.</p> 
2.1.10	<p>Click Edit identity pool link in the top-right corner.</p> 
2.1.11	<p>Copy the Identity pool ID value into the clipboard.</p> <p>Identity pool ID us-east-1:2fe60976-0538-4cd2-b973-a4cad83e99ee (Show ARN)</p> <p>You need the full Id (not the ARN), which will look like this:</p> <p>us-east-1:2fe60976-0538-4cd2-a4cad83e99ee</p> <p>Of course, your Cognito Identity Pool Id will be different to that shown above. You cannot use the Cognito Identity Pool Id shown above!</p>
2.1.12	<p>With the Cognito Identity Pool Id in your clipboard, you can now update the value in the bootcamp-config.js file</p>

2.1.13

```
var APPLIANCE_TYPE = 5;
```

APPLIANCE_TYPE

This value determines what the mobile application (which we will introduce later in the bootcamp) will show as the type of appliance – a kettle, toaster, coffee machine, etc.

Available appliance types are:

```
//  
// CONSTS  
  
define({  
    APPLIANCE_TYPE_UNKNOWN : 0,  
    APPLIANCE_TYPE_TOASTER : 1,  
    APPLIANCE_TYPE_MICROWAVE : 2,  
    APPLIANCE_TYPE_ESPRESSO : 3,  
    APPLIANCE_TYPE_BLENDER : 4,  
    APPLIANCE_TYPE_KETTLE : 5,  
});
```

Simply choose the appliance type you want to use, and set the value to the number shown above.

2.1.14

```
var BLE_UUID = "badf00d5dffb48d2b060d0f5a71096e0";
```

BLE_UUID

This value determines what the Bluetooth Low Energy Advertisement Channel will be for all BLE advertisements from your device

You can generate this value using a feature of the Central Web Site

Because you work in pairs, only one of you needs to create the value using the Central web site, and then share this value with the other member of your pair. In order for your Edison device to detect your partner's Edison device via Bluetooth, you both need to have the same BLE_UUID value

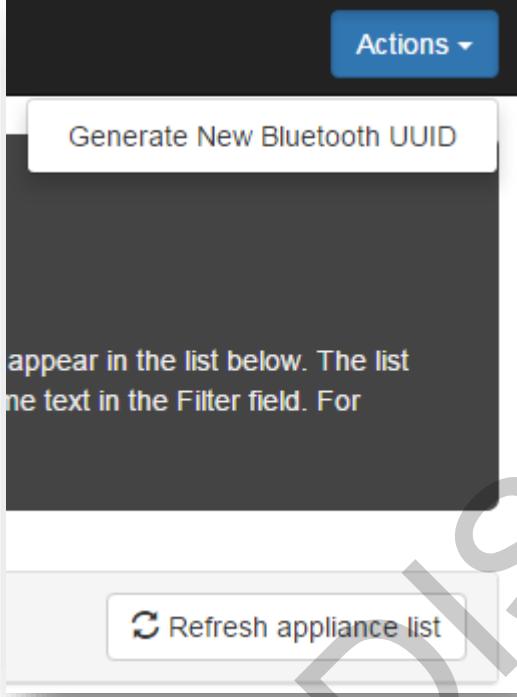
Using the Central web site to generate the BLE_UUID value:

Using your web browser, browse to the Central Web Site using this URL:

<http://bit.ly/mobile-iot-bootcamp>

The screenshot shows a web page titled "Mobile & IoT Bootcamp Central Web Site". The page header includes the AWS logo and the text "Mobile & IoT Bootcamp | re:invent 2015". A "Actions" button is in the top right corner. Below the header, there is a message: "Welcome to the Mobile & IoT Bootcamp Central Web Site for re:invent 2015. If you have configured your IoT device as per the Lab Guide instructions, your device will appear in the list below. The list is ordered by most recently registered. You can search for your appliance by entering some text in the Filter field. For example you can enter your device name, or Bluetooth UUID". There is a "Filter appliance list..." input field and a "Refresh appliance list" button. A table lists four devices:

Device Name	Type	Appliance UUID	IP Address	Bluetooth UUID	Registered
Nestor Edison	Blender	001122335566	192.168.200.22	badf00d5dffb48d2b060d0f5a71096e0	2 days ago
RTEdison1	Espresso	984fee038569	192.168.0.18	318c703978ec4a9e89727cf65b82f5b3	6 days ago
RTEdison2	Blender	984fee03de2a	192.168.0.19	318c703978ec4a9e89727cf65b82f5b3	6 days ago
Adam's Edison	Kettle	984fee033fbc	192.168.200.20	badf00d5dffb48d2b060d0f5a71096e0	7 days ago

2.1.15	<p>In the top right of the page, click the Actions menu button and choose the Generate New Bluetooth UUID option.</p>  <p>appear in the list below. The list the text in the Filter field. For</p> <p><input type="button" value="⟳ Refresh appliance list"/></p>
2.1.16	<p>The Generate New Bluetooth UUID window will appear, and a few moments later, a new UUID will be created, ready for you to use.</p> <hr/>  <p>e13e25b718d34953aa0820b12d96eab8</p> <p><input type="button" value="Close this window"/></p>

2.1.17	<p>Copy the UUID that is generated, into your clipboard. Make sure you copy the entire UUID and do not include any spaces before or after the UUID. Some browsers (such as Chrome) allow you to double-click on the text to select the entire block of text. You can then copy the text to your clipboard.</p>
2.1.18	<p>Paste the UUID you have in the clipboard into your <i>bootcamp-config.js</i> file, replacing the placeholder UUID that is pre-configured.</p> <p>Note that your partner will also need to paste the same UUID into their <i>bootcamp-config.js</i> file, to ensure they broadcast BLE packets that your device can receive - and can receive BLE packets that your device broadcasts.</p> <p>To easily make your UUID available to your partner, you can wait until your device has ‘registered’ with the Central Web Site, and then ask your partner to find your device in the list, and copy the UUID.</p> <p>You may want to come back to this step later in the process, once you have both configured your Edison devices, and then elect one of your device’s UUIDs to be the one you will both use.</p>

2.1.19

```
var GOOGLE_PROJECT_NUMBER = "0000000000";
```

GOOGLE_PROJECT_NUMBER

If you are using Android, this value is important, however if you are using OSX for our mobile app, you can skip this setting

As part of the pre-requisites for this Bootcamp, you should have already created a Google Project that we will use for our Mobile App to be able to receive push notifications.

Log into your Google developer console by browsing to:

<https://console.developers.google.com>

Select the Google project you created for use in our Bootcamp (the prerequisites guide suggested you call it Bootcamp2015)

The Project Number will be listed in the Google console. Copy it to your clipboard and paste it into your **bootcamp-config.js** file

2.1.20	<pre><code>var MOBILE_PUSH_PROVIDER = "GCM";</code></pre> <p>MOBILE_PUSH_PROVIDER</p> <p>When you set up Amazon SNS for push notifications later in the Bootcamp, you will need to select SNS endpoints for push notifications to Android, or to iOS. The type of notification service you are using is encoded into the SNS endpoint ARN, and the value of MOBILE_PUSH_PROVIDER is used to construct these ARNs.</p> <p><i>For iOS:</i></p> <p>Set the value of MOBILE_PUSH_PROVIDER to be APNS_SANDBOX</p> <pre><code>var MOBILE_PUSH_PROVIDER = "APNS_SANDBOX";</code></pre> <p><i>For Android:</i></p> <p>Set the value of MOBILE_PUSH_PROVIDER to be GCM</p> <pre><code>var MOBILE_PUSH_PROVIDER = "GCM";</code></pre>
2.1.21	<p>Save the changes that you modified with your text editor. If you are using vi this can be done by pressing the 'esc' (escape) key, typing :x and then pressing enter. You should then see the prompt return to the main terminal screen.</p>

2.1.22	<p>Now we will add a ‘service’ that will be automatically run at startup on the Edison</p> <p>The service will automatically register your Edison device against the Bootcamp Central website, which will be used in the Bootcamp to tether the mobile companion app to your Edison, and make it easier to track the IP address your Edison is using (which can change after a boot because the addresses are allocated using DHCP)</p>
2.1.23	<p>For Mac OSX users:</p> <p>Now that our Edison has network activity we will disconnect from the Putty Serial (Windows) or SCREEN (Mac) session and connect via SSH.</p> <p>Press ‘Ctrl-a’ and then the ‘d’ key as seen below (or Ctrl-d).</p>  <p>You should then be back to your main terminal before you opened up a screen session. If you are on a Mac type ‘ssh root@IPADDRESS’ where IPADDRESS is the value you noted on the Edison before when configuring the WiFi. Press Enter, accept the security fingerprint and then enter your password to login as root to your Edison.</p>

2.1.24

For Windows users:

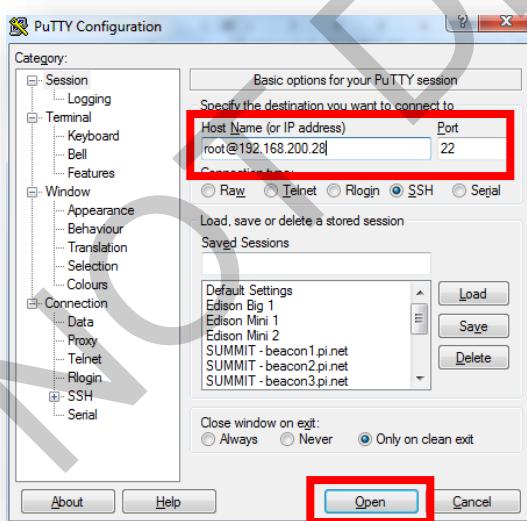
You can simply close the serial terminal session window (you may be prompted to confirm) to disconnect from the Edison. Then launch another new PuTTY window.

In the **Host Name (or Ip address)** field, enter root@ and the IP address of your Edison (the value you noted down on the Edison before when configuring the WiFi).



In the above example, the IP address of the Edison is 192.168.200.28 so the value entered in the Host Name is:

root@192.168.200.28



Of course, your IP address will be different! You cannot use the address above. Replace it with the IP address of your Edison, prefixed by root@

Make sure SSH is selected under **Connection type** which should be the default.

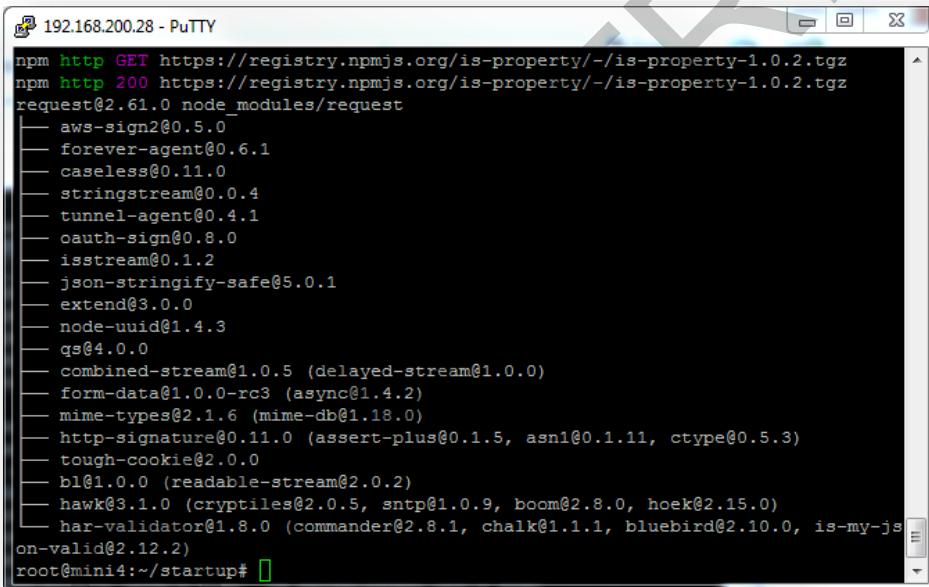
Click **Open** and when prompted, provide the password you set previously. If you followed the recommendation, the password will be **edison2015**

You may also be asked to confirm the fingerprint of the device. You should accept this and continue.

- 2.1.25 From the SSH session you have connected to your Edison, cd in to the /home/root/startup folder on your Edison and install all the NodeJS dependencies needed to run the startup process, by entering the following commands:

```
cd /home/root/startup  
npm install
```

You will see the following output in the SSH console – ignore any warnings you may see, these will be for optional components we won't be using:



The screenshot shows a PuTTY terminal window titled "192.168.200.28 - PuTTY". The window displays the output of an "npm install" command. The output shows various Node.js packages being downloaded from the npm registry. Some packages are listed with their version numbers and descriptions, such as "request@2.61.0 node_modules/request", "aws-sign2@0.5.0", and "forever-agent@0.6.1". The terminal ends with the prompt "root@mini4:~/startup#".

When this completes, run the following commands:

```
mv /home/root/home-root/labs /home/root/.  
cd /home/root/labs  
npm install
```

This will move the lab source code into the appropriate folder and then ensure that the Node package dependencies for startup and the various Bootcamp software labs have all been installed and are ready to use – ignore any warnings you may see, these will be for optional components we won't be using

2.1.26	<p>We have created a ‘service’ to run at Edison startup, which will automatically register your Edison with our Central web site. In the next steps, we will set this up.</p> <p>Run the following command to move the service definition into the correct location on your Edison:</p> <pre>mv /home/root/lib-systemd-system/bootcamp-register.service /lib/systemd/system</pre>					
2.1.27	<p>Now run the following to register the service:</p> <pre>systemctl enable bootcamp-register.service systemctl start bootcamp-register.service</pre>					
2.1.28	<p>To confirm all is well, browse to the Bootcamp Central web site to ensure your Edison has registered and appears in the list. Open your browser and browse to:</p> <p>http://bit.ly/mobile-iot-bootcamp</p>					
2.1.29	<p>You should see your device in the list, like in the image shown below:</p>  <table border="1"> <tr> <td>Adam's Edison</td> <td>Kettle</td> <td>984fee033efbc</td> <td>192.168.200.20</td> <td>badf00d5dfffb48d2b060d0f5a71096e0</td> </tr> </table>	Adam's Edison	Kettle	984fee033efbc	192.168.200.20	badf00d5dfffb48d2b060d0f5a71096e0
Adam's Edison	Kettle	984fee033efbc	192.168.200.20	badf00d5dfffb48d2b060d0f5a71096e0		
2.1.30	<p>We have now finished installing the correct files on our Edison, so we should now clean up our home directory. Issue these commands to do so:</p> <pre>rm -R /home/root/home-root/ rm -R /home/root/etc/ rm -R /home/root/Edison.tar rm -R /home/root/0.10.28/ rm -R /home/root/lib-systemd-system/ rm -R /home/root/tmp/</pre>					

2.1.31	Now our home directory is clean except for the files we will need ongoing, on the Edison, Reboot the device by issuing: reboot
2.1.32	When the Edison comes back up, SSH on to it (following the steps described previously – don't forget to add root@ before your IP address!) and confirm you can connect. You should check the Central site for the IP address of your Edison device, just in case it has been re-assigned by DHCP. Remember, you will need to use root as the username, and the password you set earlier when you configured your Edison in an earlier step

2.1.33 Finally, remember that you and your partner need to both be using the same UUID for the **BLE_UUID** configuration value in the *bootcamp-config.js* file on your respective Edison devices. It does not matter who's generated UUID you use, but you both need to be using the same in order for your devices to receive each other's BLE packets

Decide between you which UUID you will use. The other student should then browse the Central Web Site using this URL:

<http://bit.ly/mobile-iot-bootcamp>

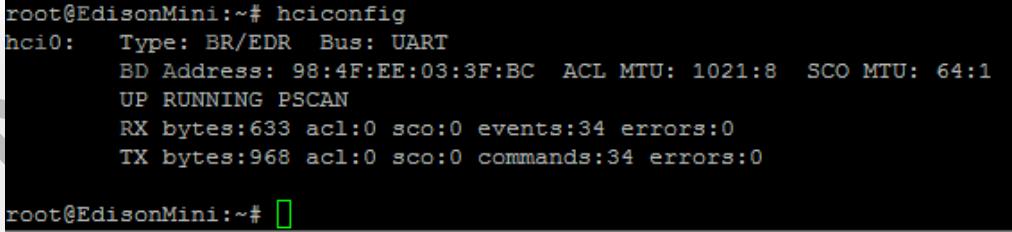
The screenshot shows a web page titled "Mobile & IoT Bootcamp Central Web Site" from re:invent 2015. The page features the Amazon Web Services logo. A message at the top says: "Welcome to the Mobile & IoT Bootcamp Central Web Site for re:invent 2015. If you have configured your IoT device as per the Lab Guide instructions, your device will appear in the list below. The list is ordered by most recently registered. You can search for your appliance by entering some text in the Filter field. For example you can enter your device name, or Bluetooth UUID." Below this is a table with the following data:

Device Name	Type	Appliance UUID	IP Address	Bluetooth UUID	Registered
Nestor Edison	Blender	001122335566	192.168.200.22	badf00d5dfffb48d2b060d0f5a71096e0	2 days ago
RTEdison1	Espresso	984fee038569	192.168.0.18	318c703978ec4a9e89727cf65b82ff5b3	6 days ago
RTEdison2	Blender	984fee03de2a	192.168.0.19	318c703978ec4a9e89727cf65b82ff5b3	6 days ago
Adam's Edison	Kettle	984fee033fbc	192.168.200.20	badf00d5dfffb48d2b060d0f5a71096e0	7 days ago

Use the Filter field to locate your partner's device by name, and then copy the **Bluetooth UUID** value to your *bootcamp-config.js* file on your Edison.

Task 2-2: Optional: Confirming Bluetooth LE is functioning on your device

Overview	<p><i>This task is optional. It is not necessary for you to complete this section, because it just confirms and validates that your device is functioning, and there are no changes made during this optional task.</i></p> <p>We will now test the Bluetooth module on your Edison device.</p>
----------	---

Step	Instruction
2.2.1	<p>Navigate to /home/root/ by entering</p> <pre>cd /home/root/</pre>
2.2.2	<p>Confirm Bluetooth is running by issuing the <i>hciconfig</i> command.</p> <pre>hciconfig</pre>  <pre>root@EdisonMini:~# hciconfig hci0: Type: BR/EDR Bus: UART BD Address: 98:4F:EE:03:3F:BC ACL MTU: 1021:8 SCO MTU: 64:1 UP RUNNING PSCAN RX bytes:633 acl:0 sco:0 events:34 errors:0 TX bytes:968 acl:0 sco:0 commands:34 errors:0 root@EdisonMini:~# [green square]</pre> <p>Here you can see the MAC address of this unit is 98:4F:EE:03:3F:BC and it is up and running</p>

2.2.3 Check that the hci0 interface is happy by issuing

```
hciconfig hci0 lestates
```

```
root@EdisonArduino:~/node_app_slot# hciconfig hci0 lestates
Supported link layer states:
        YES Non-connectable Advertising State
        YES Scannable Advertising State
        YES Connectable Advertising State
        YES Directed Advertising State
        YES Passive Scanning State
        YES Active Scanning State
        YES Initiating State/Connection State in Master Role
        YES Connection State in the Slave Role
        YES Non-connectable Advertising State and Passive Scanning State combination
        YES Scannable Advertising State and Passive Scanning State combination
        YES Connectable Advertising State and Passive Scanning State combination
        YES Directed Advertising State and Passive Scanning State combination
        YES Non-connectable Advertising State and Active Scanning State combination
        YES Scannable Advertising State and Active Scanning State combination
        YES Connectable Advertising State and Active Scanning State combination
        YES Directed Advertising State and Active Scanning State combination
        YES Non-connectable Advertising State and Initiating State combination
        YES Scannable Advertising State and Initiating State combination
        YES Non-connectable Advertising State and Master Role combination
        YES Scannable Advertising State and Master Role combination
        YES Non-connectable Advertising State and Slave Role combination
        YES Scannable Advertising State and Slave Role combination
        YES Passive Scanning State and Initiating State combination
        YES Active Scanning State and Initiating State combination
        YES Passive Scanning State and Master Role combination
        YES Active Scanning State and Master Role combination
        YES Passive Scanning State and Slave Role combination
        YES Active Scanning State and Slave Role combination
        YES Initiating State and Master Role combination/Master Role and Master
Role combination
root@EdisonArduino:~/node_app_slot#
```

2.2.4 See what devices are broadcasting LE by issuing:

```
hcitool lescan
```

NOTE: This step uses hcitool not hciconfig!

```
root@EdisonArduino:~/node_app_slot# hcitool lescan
LE Scan ...
E2:EE:E7:B2:42:78 (unknown)
E2:EE:E7:B2:42:78 (unknown)
FF:8A:10:C9:29:C8 (unknown)
EC:1D:41:12:85:71 (unknown)
EC:1D:41:12:85:71 (unknown)
FF:8A:10:C9:29:C8 (unknown)
```

2.2.5	<p>If you can see other devices, it means your Bluetooth stack is working as expected.</p> <p>If you do not see any devices listed under the 'LE Scan' your Bluetooth may be working fine, but there are no other devices in range.</p> <p>As long as you do not get any errors, you can continue assuming the BLE stack is running fine.</p>
-------	---

Lab 4: Writing BLE detections to Amazon Kinesis Stream

Overview	In this module, we will use the Edison to detect Bluetooth Low Energy Advertisement packets emitted by your partner's Edison device, and to write the detection data to the Amazon Kinesis stream. We first do this at high-velocity, sending every BLE detection to the stream, and then refine our approach to only send data to the stream once per second.
Objectives	After completing this lab, you will be able to: <ul style="list-style-type: none">Understand how we can use Amazon Kinesis for high-velocity telemetryUse AWS CloudWatch metrics to monitor the throughput of the Amazon Kinesis stream.
Pre-requisites	This lab requires: <ul style="list-style-type: none">Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">Note: The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.For Microsoft Windows users: Administrator access to the computer.An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).

Duration	15 minutes
-----------------	------------

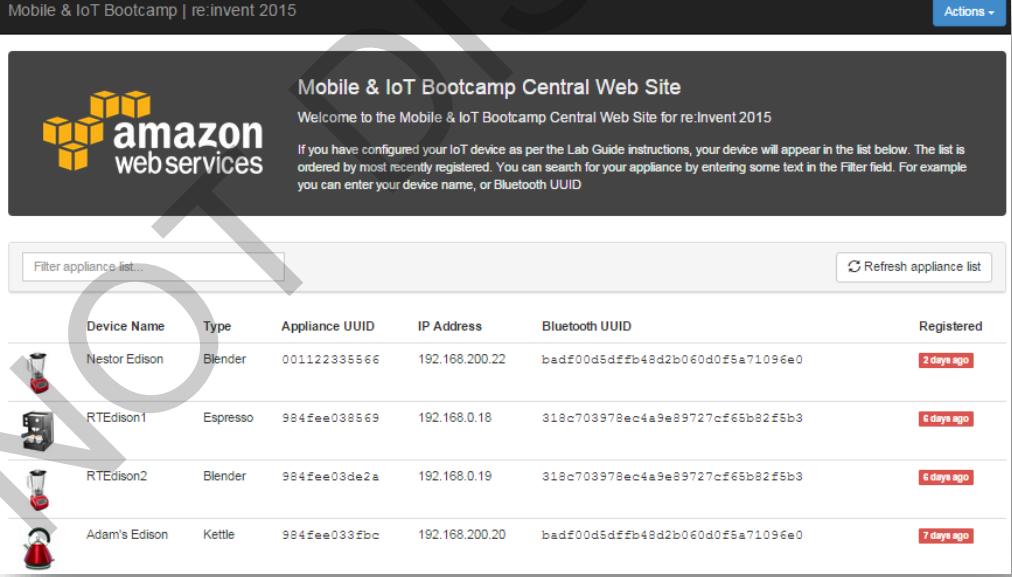
DO NOT DISTRIBUTE

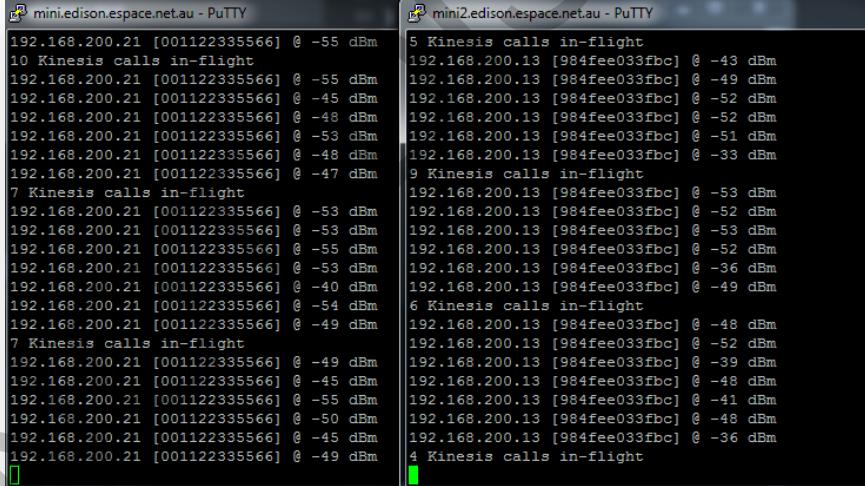
Task 1: Streaming BLE data to an Amazon Kinesis Stream

Overview	In this module, we will detect the Bluetooth Low Energy Advertisement packets on your Edison device, emitted from your partner's Edison device. We will write the detection payload to our system's Amazon Kinesis stream.
-----------------	--

Task 1-1: Stream BLE Data to Amazon Kinesis

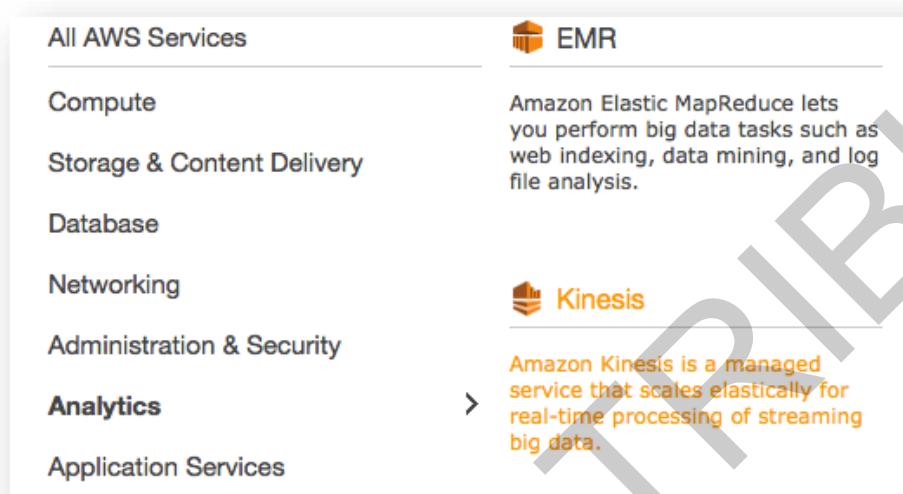
Overview	We will connect to our Edison device using the SSH protocol, then run a test script that sends BLE advertisements that are detected, to our Amazon Kinesis stream, and monitor the throughput using AWS CloudWatch
-----------------	--

Step	Instruction																														
1.1.1	<p>Connect to your Edison device using the SSH protocol.</p> <p>Windows users Use the Putty application to connect to your Edison on the IP address shown in the Central web site for your device. Don't forget to add root@ before the IP address!</p> <p>Mac users Use the terminal to SSH to your Edison on the IP address shown in the Central web site for your device. Don't forget to add root@ before the IP address!</p> <p>You can browse to the Central web site and determine the IP address for your device. Go to the following location in your browser:</p> <p>http://bit.ly/mobile-iot-bootcamp</p>  <table border="1"> <thead> <tr> <th>Device Name</th> <th>Type</th> <th>Appliance UUID</th> <th>IP Address</th> <th>Bluetooth UUID</th> <th>Registered</th> </tr> </thead> <tbody> <tr> <td>Nestor Edison</td> <td>Blender</td> <td>001122335566</td> <td>192.168.200.22</td> <td>badf00d5dffeb48d2b060d0f5a71096e0</td> <td>2 days ago</td> </tr> <tr> <td>RTEdison1</td> <td>Espresso</td> <td>984f8ee038569</td> <td>192.168.0.18</td> <td>318c703978ec4a9e89727cf65b82f5b3</td> <td>6 days ago</td> </tr> <tr> <td>RTEdison2</td> <td>Blender</td> <td>984f8ee03de2a</td> <td>192.168.0.19</td> <td>318c703978ec4a9e89727cf65b82f5b3</td> <td>6 days ago</td> </tr> <tr> <td>Adam's Edison</td> <td>Kettle</td> <td>984f8ee033fbc</td> <td>192.168.200.20</td> <td>badf00d5dffeb48d2b060d0f5a71096e0</td> <td>7 days ago</td> </tr> </tbody> </table> <p>Locate your device by Name, and determine your IP address. You can use the Filter feature to enter in the name of your device to help locate your Edison in the list</p>	Device Name	Type	Appliance UUID	IP Address	Bluetooth UUID	Registered	Nestor Edison	Blender	001122335566	192.168.200.22	badf00d5dffeb48d2b060d0f5a71096e0	2 days ago	RTEdison1	Espresso	984f8ee038569	192.168.0.18	318c703978ec4a9e89727cf65b82f5b3	6 days ago	RTEdison2	Blender	984f8ee03de2a	192.168.0.19	318c703978ec4a9e89727cf65b82f5b3	6 days ago	Adam's Edison	Kettle	984f8ee033fbc	192.168.200.20	badf00d5dffeb48d2b060d0f5a71096e0	7 days ago
Device Name	Type	Appliance UUID	IP Address	Bluetooth UUID	Registered																										
Nestor Edison	Blender	001122335566	192.168.200.22	badf00d5dffeb48d2b060d0f5a71096e0	2 days ago																										
RTEdison1	Espresso	984f8ee038569	192.168.0.18	318c703978ec4a9e89727cf65b82f5b3	6 days ago																										
RTEdison2	Blender	984f8ee03de2a	192.168.0.19	318c703978ec4a9e89727cf65b82f5b3	6 days ago																										
Adam's Edison	Kettle	984f8ee033fbc	192.168.200.20	badf00d5dffeb48d2b060d0f5a71096e0	7 days ago																										

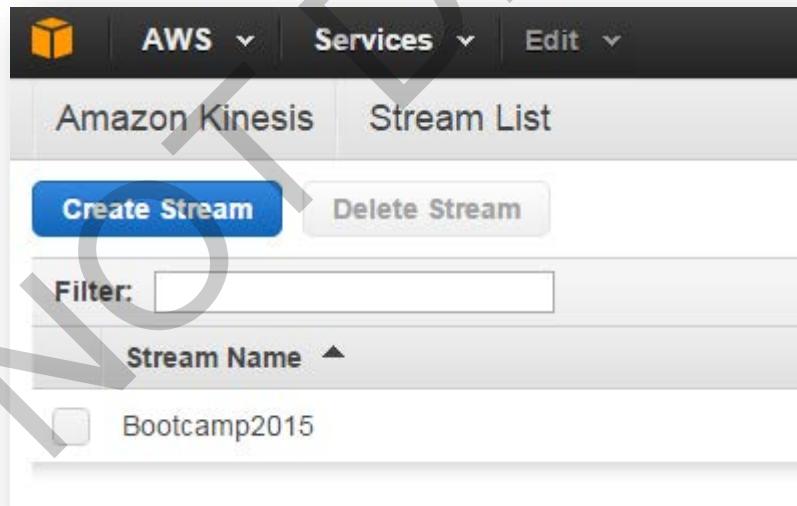
1.1.2	<p>You will work in pairs for this lab, so ensure that you and your partner are using the same UUID value in your bootcamp-config.js file.</p> <p>Refer to lab 3 for details on how to set this up if you haven't done so.</p>
1.1.3	<p>When you are connected to your Edison via SSH, cd into the labs folder and run the BLE test script by issuing the following commands:</p> <pre>cd /home/root/labs node test-ble-to-kinesis.js</pre>
1.1.4	<p>If there are other Edison's in range and configured to use the same BLE UUID in their /etc/bootcamp2015-config.js configuration file, and are also running this test file, you will see something similar to the below.</p> <p>In this example, the Edison with MAC 001122335566 at IP address 192.168.200.21 is hearing BLE advertisements from the Edison with MAC address 984fee033fbc and IP address 192.168.200.13 – and vice-versa.</p>  <pre>miniedison.espace.net.au - PuTTY 192.168.200.21 [001122335566] @ -55 dBm 10 Kinesis calls in-flight 192.168.200.21 [001122335566] @ -55 dBm 192.168.200.21 [001122335566] @ -45 dBm 192.168.200.21 [001122335566] @ -48 dBm 192.168.200.21 [001122335566] @ -53 dBm 192.168.200.21 [001122335566] @ -48 dBm 192.168.200.21 [001122335566] @ -47 dBm 7 Kinesis calls in-flight 192.168.200.21 [001122335566] @ -53 dBm 192.168.200.21 [001122335566] @ -53 dBm 192.168.200.21 [001122335566] @ -55 dBm 192.168.200.21 [001122335566] @ -53 dBm 192.168.200.21 [001122335566] @ -40 dBm 192.168.200.21 [001122335566] @ -54 dBm 192.168.200.21 [001122335566] @ -49 dBm 7 Kinesis calls in-flight 192.168.200.21 [001122335566] @ -49 dBm 192.168.200.21 [001122335566] @ -45 dBm 192.168.200.21 [001122335566] @ -55 dBm 192.168.200.21 [001122335566] @ -50 dBm 192.168.200.21 [001122335566] @ -45 dBm 192.168.200.21 [001122335566] @ -49 dBm mini2.edison.espace.net.au - PuTTY 5 Kinesis calls in-flight 192.168.200.13 [984fee033fbc] @ -43 dBm 192.168.200.13 [984fee033fbc] @ -49 dBm 192.168.200.13 [984fee033fbc] @ -52 dBm 192.168.200.13 [984fee033fbc] @ -52 dBm 192.168.200.13 [984fee033fbc] @ -51 dBm 192.168.200.13 [984fee033fbc] @ -33 dBm 9 Kinesis calls in-flight 192.168.200.13 [984fee033fbc] @ -53 dBm 192.168.200.13 [984fee033fbc] @ -52 dBm 192.168.200.13 [984fee033fbc] @ -53 dBm 192.168.200.13 [984fee033fbc] @ -52 dBm 192.168.200.13 [984fee033fbc] @ -36 dBm 192.168.200.13 [984fee033fbc] @ -49 dBm 6 Kinesis calls in-flight 192.168.200.13 [984fee033fbc] @ -48 dBm 192.168.200.13 [984fee033fbc] @ -52 dBm 192.168.200.13 [984fee033fbc] @ -39 dBm 192.168.200.13 [984fee033fbc] @ -48 dBm 192.168.200.13 [984fee033fbc] @ -41 dBm 192.168.200.13 [984fee033fbc] @ -48 dBm 192.168.200.13 [984fee033fbc] @ -36 dBm 4 Kinesis calls in-flight</pre>

The screen shot above shows the output from two SSH sessions, one to each Edison that is running the test. If you are working with your partner, you will have one SSH session, and your partner will have another.

- 1.1.5 Login to the AWS Management Console from your Qwiklabs Environment and choose **Services - Analytics - Kinesis**.

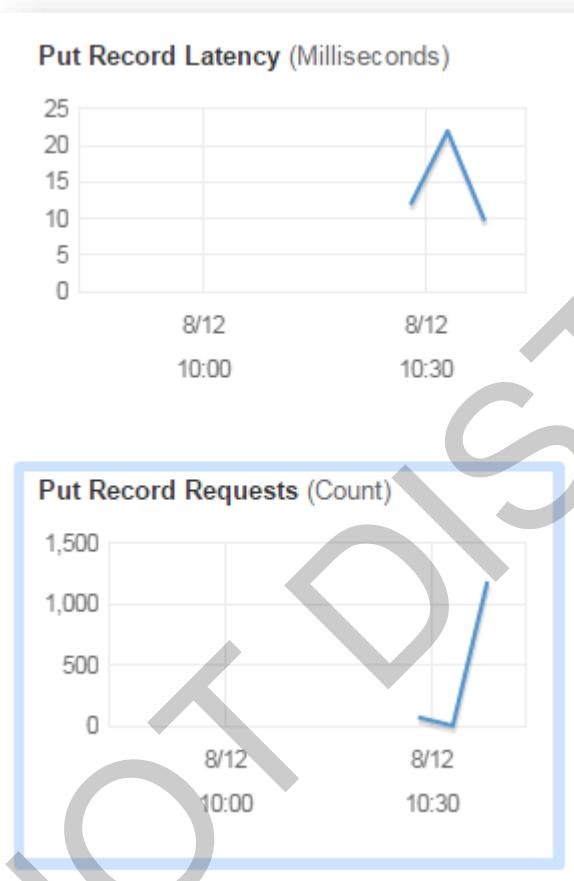


- 1.1.6 Click the **Bootcamp2015** link.



- 1.1.7 Scroll down to the monitoring section where you can view the throughput to your stream.

Note that there may be a few minutes of delay before you see any data points appear in the CloudWatch metrics. This is normal behavior.



- 1.1.8 Notice how the Put throughput is quite high. Click on the **Put Record Requests (Count)** box to view the metric in more detail:



We don't need our BLE detections to be sent with such high velocity. Once per second is enough resolution for our needs. In the next section, we will update our test script to address this requirement.

Press 'Ctrl-c' to stop the test-ble-to-kinesis.js code from running and move onto the next section.

Task 2: Streaming BLE data to the Amazon Kinesis Stream more efficiently

Overview	In this task, we build on the previous task of listening for BLE advertisements, and writing those detected to the Amazon Kinesis stream. This task uses a version that is more efficient - batching up detections and sending data to Amazon Kinesis once per second rather than 'continually'.
----------	--

Step	Instruction
2.1.1	Run the test script using the standard NodeJS process: <pre>cd /home/root/labs node test-ble-to-kinesis-smoothed-and-batched.js</pre>

- 2.1.2 The output from running the script will look like the image shown below, if other Edisons are in range, using the same UUID as your Edison is configured to use, and running one of the scripts in the labs that advertises data via BLE (for example, your partner is running this lab's script as well):

```
Initialising Cognito...
Cognito IdentityId => us-east-1:a822ef27-bef2-4d44-bee2-824d8eda083c
Starting to listen for other Edisons...
New Edison discovered: 984fee033fbc @ -67
192.168.200.21:5(984fee033fbc) :: 0 dBm >> 0 metres away
192.168.200.21:5(984fee033fbc) :: -65.4646 dBm >> 1.4167 metres away
192.168.200.21:5(984fee033fbc) :: -70.0920 dBm >> 1.9431 metres away
192.168.200.21:5(984fee033fbc) :: -69.1324 dBm >> 1.8159 metres away
192.168.200.21:5(984fee033fbc) :: -70.6612 dBm >> 2.0236 metres away
192.168.200.21:5(984fee033fbc) :: -70.2384 dBm >> 1.9634 metres away
192.168.200.21:5(984fee033fbc) :: -71.5096 dBm >> 2.1510 metres away
192.168.200.21:5(984fee033fbc) :: -69.7183 dBm >> 1.8923 metres away
192.168.200.21:5(984fee033fbc) :: -71.6891 dBm >> 2.1792 metres away
192.168.200.21:5(984fee033fbc) :: -72.2067 dBm >> 2.2627 metres away
192.168.200.21:5(984fee033fbc) :: -71.4233 dBm >> 2.1376 metres away
192.168.200.21:5(984fee033fbc) :: -69.3985 dBm >> 1.8501 metres away
192.168.200.21:5(984fee033fbc) :: -68.2821 dBm >> 1.7116 metres away
192.168.200.21:5(984fee033fbc) :: -68.4026 dBm >> 1.7259 metres away
192.168.200.21:5(984fee033fbc) :: -68.3058 dBm >> 1.7144 metres away
192.168.200.21:5(984fee033fbc) :: -69.3238 dBm >> 1.8404 metres away
192.168.200.21:5(984fee033fbc) :: -69.3483 dBm >> 1.8436 metres away
192.168.200.21:5(984fee033fbc) :: -68.5069 dBm >> 1.7384 metres away
192.168.200.21:5(984fee033fbc) :: -71.0326 dBm >> 2.0783 metres away
```

Notice how the output appears only once per second, unlike before where the data was being sent more frequently.

Also notice that the output shows the RSSI as well as the calculated distance.

In this screenshot, the Primary and Secondary Edisons were sitting close to each other, hence the low distance calculation.

2.1.3	<p>You should spend a few moments looking through the source code for this lab, since it adds the Kalman filtering and calculation distance functionality to our project.</p> <p>You can access the lab source code on the Edison in the /home/root/labs folder, or if you prefer, download the lab source code to your computer. You did this in an earlier lab by browsing to this link:</p> <p>http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/labs.zip</p>
2.1.4	<p>Press 'Ctrl-c' to stop the test-ble-to-kinesis-smoothed-and-batched.js code from running and move onto the next section.</p>

Lab 5: Using AWS Lambda to write BLE detection data to Amazon DynamoDB

Overview	<p>In this lab, we introduce AWS Lambda. Previously, we wrote BLE detection data to an Amazon Kinesis stream, and in this lab, we bind to the stream with a Lambda function that processes the incoming records and writes the results into an Amazon DynamoDB table that is used to record the BLE detection data and make it available to the Mobile App</p>
Objectives	<p>After completing this lab, you will be able to:</p> <ul style="list-style-type: none">• Create an AWS Lambda function that is triggered by a record arriving on an Amazon Kinesis stream• Monitor AWS CloudWatch logs for AWS Lambda output• Create an Amazon DynamoDB table• Update records in Amazon DynamoDB using NodeJS
Pre-requisites	<p>This lab requires:</p> <ul style="list-style-type: none">• Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">○ Note: The <i>qwikLABS</i> lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.• For Microsoft Windows users: Administrator access to the computer.• An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).

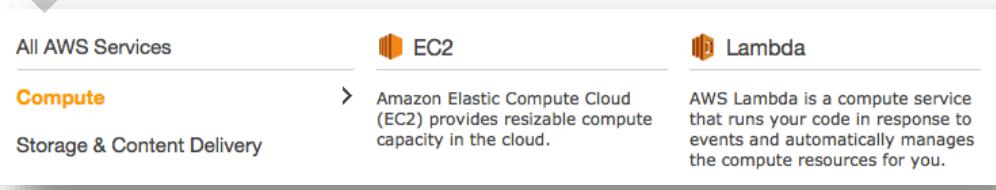
Duration	20 minutes
-----------------	------------

Task 1: Creating an AWS Lambda function

Overview	In this section we create the actual AWS Lambda function that we will use to collect data from the Amazon Kinesis Stream. We then test out this Lambda function by streaming the BLE data from our previous labs into Amazon Kinesis and have the Lambda function process these events as they are streamed.
-----------------	--

Task 1-1: Create the Lambda function

Overview	In this section we will setup and create the Lambda function that we will be using for this exercise.
-----------------	---

Step	Instruction
1.1.1	<p>Launch the AWS Management Console from your Qwiklabs environment and click Lambda under Compute from the drop-down menu.</p>  <p>The screenshot shows the AWS Management Console navigation bar. The 'Compute' option is highlighted in orange, indicating it is selected. Below the navigation bar, there are three main service links: 'All AWS Services', 'EC2' (with a small orange cube icon), and 'Lambda' (with a small orange hexagon icon). A tooltip for 'Lambda' provides a brief description: 'AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.'</p>

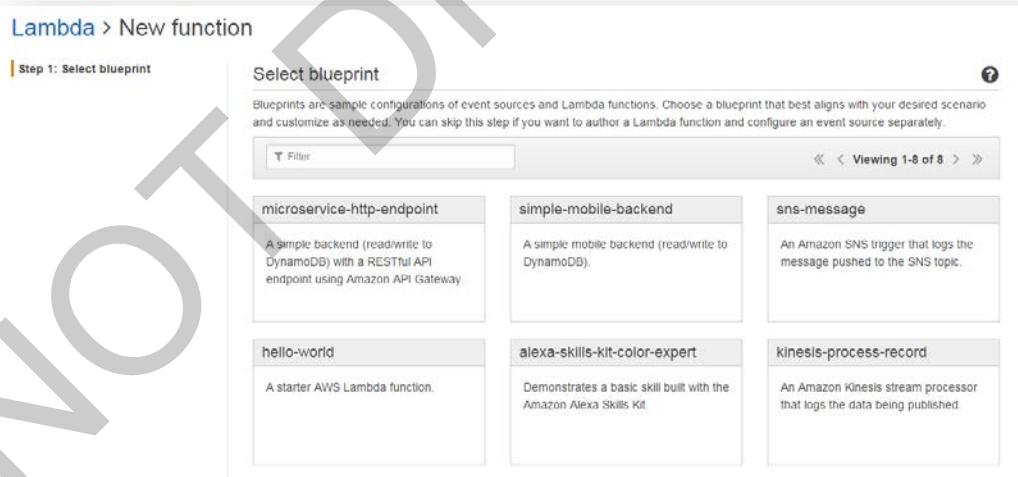
1.1.2

Click the **Get Started Now** button.



1.1.3

You will see the Blueprint panel. Select the **Kinesis-Process-Record** option.



- 1.1.4 On the **Configure Event Sources** page, select our Kinesis stream from the drop-down list, and also choose **Latest** as the Starting Position.

Configure event sources

Choose the appropriate event source for your Lambda function.

Event source type	Kinesis
Kinesis stream	Bootcamp2015
Batch size	100
Starting position	Latest

Click the **Next** button.

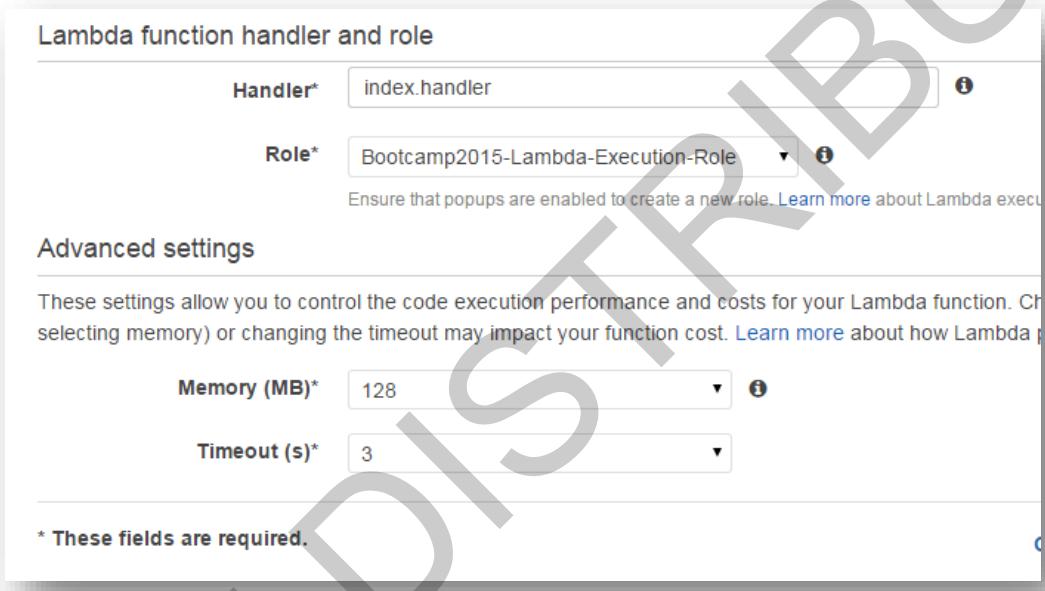
- 1.1.5 Give the Lambda function a name.

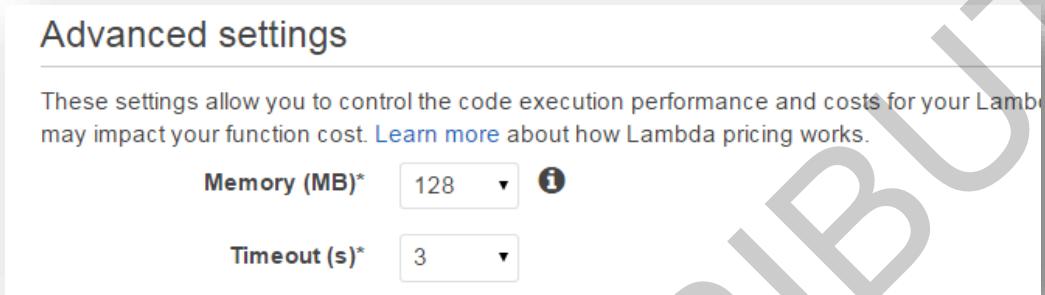
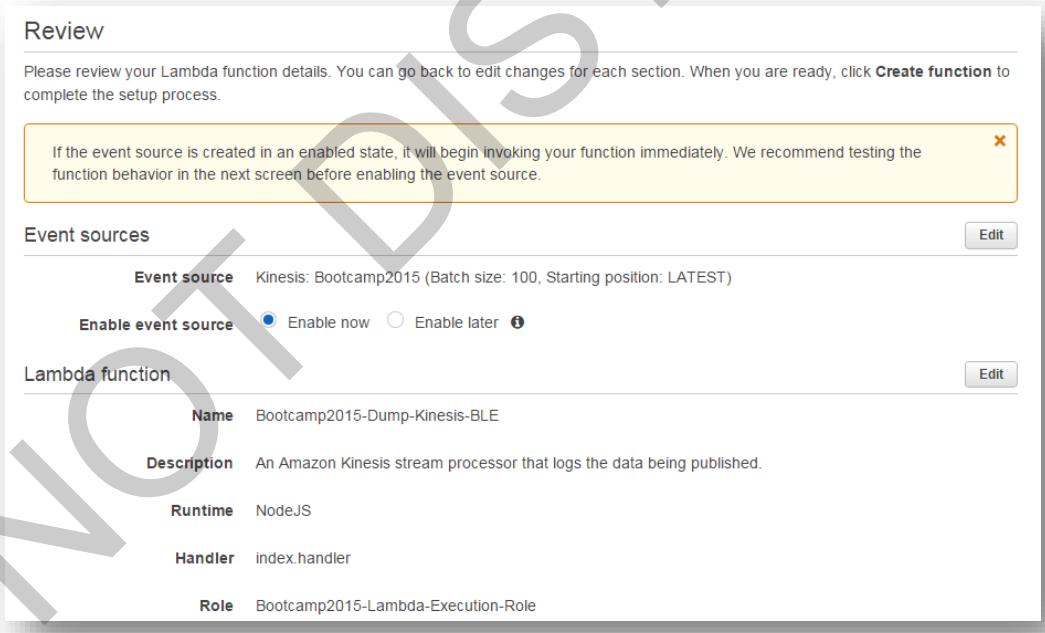
Bootcamp2015-Dump-Kinesis-BLE

Configure function

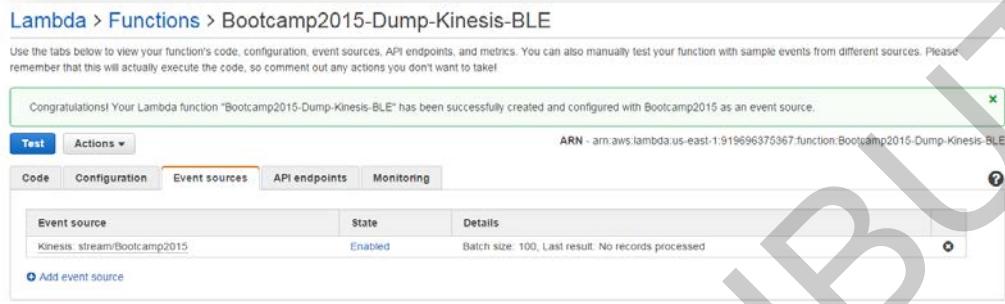
A Lambda function consists of the custom code you want to execute. [Learn more about Lambda functions.](#)

Name*	Bootcamp2015-Dump-Kinesis-BLE
Description	An Amazon Kinesis stream processor that logs the data bein
Runtime*	Node.js

1.1.6	Leave the rest of the options as they are, including the source code, which is boilerplate code that will simply send received records to AWS CloudWatch Logs.
1.1.7	Scroll down below the code section, to the Lambda function handler and role section.
	 <p>The screenshot shows the 'Lambda function handler and role' configuration page. The 'Handler' field is set to 'index.handler'. The 'Role*' field is set to 'Bootcamp2015-Lambda-Execution-Role'. Below these fields, a note says 'Ensure that popups are enabled to create a new role.' A link to 'Learn more about Lambda execution roles' is provided. Under 'Advanced settings', there are dropdown menus for 'Memory (MB)*' (set to 128) and 'Timeout (s)*' (set to 3). A note at the bottom states '* These fields are required.'</p>
1.1.8	Choose the Bootcamp2015-Lambda-Execution-Role we created earlier from the Role list.

	<p>1.1.9 Leave the Advanced settings as they are; they should be adequate for this module.</p> 
	<p>1.1.10 Click the Next button to review the function setup.</p> 
	<p>1.1.11 Click the Enable Now option in the Enable event source section. This will ensure that new Amazon Kinesis records are automatically set to invoke the new Lambda function.</p>

1.1.12 The Lambda function will be created, and you will see a success message.

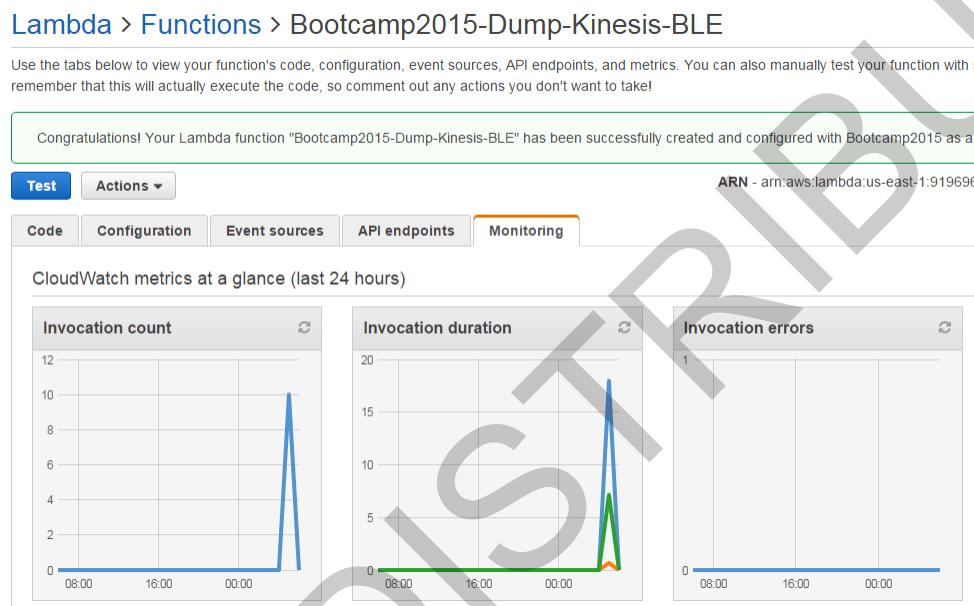


Task 1-2: Send BLE data to the Kinesis stream

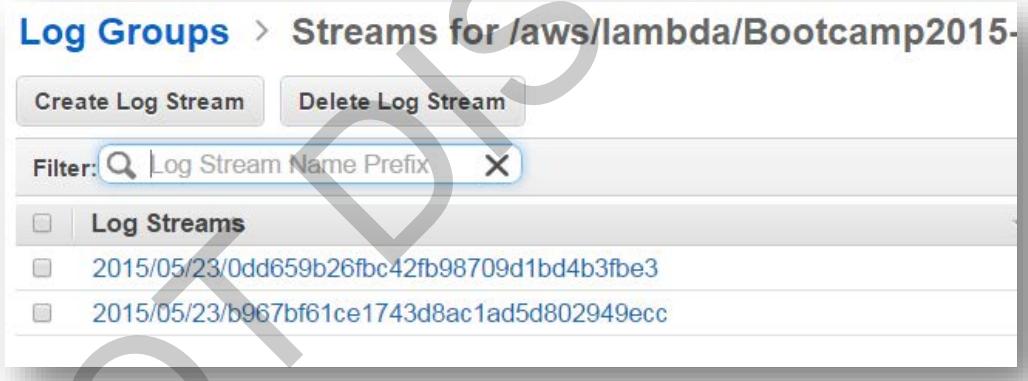
Overview We now need to send data to the stream, and observe the log data from the Lambda function.

Step	Instruction
1.2.1	<p>On your Edison, run</p> <pre>cd /home/root/labs node test-ble-to-kinesis-smoothed-and-batched</pre> <p>This is the same program we ran earlier when we wanted to push BLE detection data up to Kinesis from the Edison. This time when we run it, now that we have the Lambda function in place, we will be able to view the logs with the data, as per the code in the Lambda function.</p> <p>Note: You will see activity on the Edison console only if there is at least one other Edison in range, that uses the same broadcast channel that your Edison is using, based on the bootcamp-config.js file.</p>

- 1.2.2 Go back to the AWS Lambda dashboard - within the Lambda function - and click the **Monitoring** tab. In a few moments, you will notice that the **Invocation Count** and **Duration** CloudWatch metrics will have data.



Note: It may take a minute or two before the graphs update. You do not need to wait for them to update before continuing; you can come back to this step and refresh the view later to see if the graphs have updated.

1.2.3	<p>You can see that the Lambda function is being called. Click the View in CloudWatch link to view the logs.</p> 
1.2.4	<p>Choose one of the links, and the events will be displayed.</p> 

1.2.5	<p>Browse through the data. You will see the records being dumped out as they arrive, together with metrics regarding RAM usage and duration of execution.</p> <pre> ▼ START RequestId: 7906eb42-b99c-4091-9bcf-16d0dba16934 ▼ 2015-08-12T11:35:17.744Z 7906eb42-b99c-4091-9bcf-16d0dba16934 Decoded payload: { "rssi": 0, "confidence": 0, "filteredRSSI": 0, "calculatedDistance": 0, "lastDetectionTimestamp": "2015-08-12T11:35:15.608Z", "arrayDetections": [{ "name": "192.168.200.22:1", "uuid": "001122335566", "rssi": -67, "confidence": 36.37, "filteredRSSI": "-69.3795", "calculatedDistance": "1.8476", "lastDetectionTimestamp": "2015-08-12T11:35:15.529Z", "arrayDetections": null }] } ▼ END RequestId: 7906eb42-b99c-4091-9bcf-16d0dba16934 ▼ REPORT RequestId: 7906eb42-b99c-4091-9bcf-16d0dba16934 Duration: 0.56 ms Billed Duration: 100 ms </pre>
1.2.6	In the next section, we will write the detection data into a DynamoDB table.
1.2.7	Press 'Ctrl-c' to stop the test-ble-to-kinesis-smoothed-and-batched.js code from running and move onto the next section.

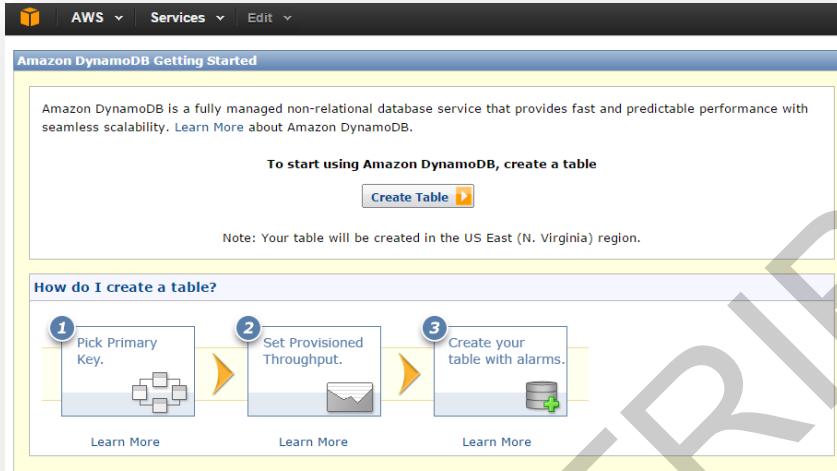
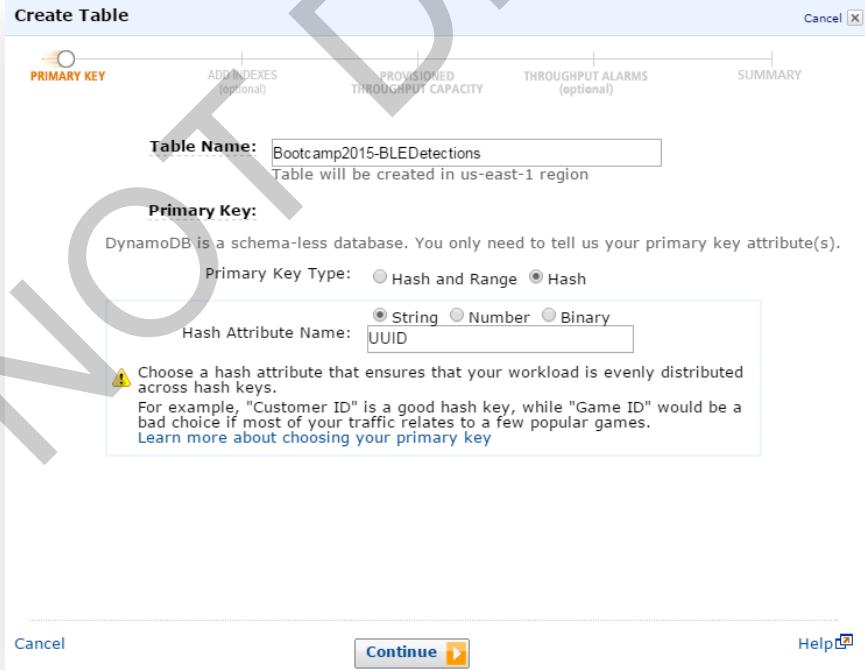
Task 2: Creating a DynamoDB Table

Overview	In this section we will create a DynamoDB table that will hold the BLE data being transmitted from your Edisons.
-----------------	--

Task 2-1: Create a DynamoDB Table

Overview	We will create a DynamoDB table and assign the appropriate throughput and table structure to store the BLE data.
-----------------	--

Step	Instruction
2.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Database – DynamoDB.</p> <p>All AWS Services</p> <p>Compute</p> <p>Storage & Content Delivery</p> <p>Database</p> <p>Networking</p> <p>Administration & Security</p> <p>Analytics</p> <p>Application Services</p> <p>The screenshot shows the AWS Management Console navigation bar. The 'Database' section is highlighted in blue, indicating it is the current category. Below it, the 'DynamoDB' service is also highlighted in blue. To the left of the navigation bar, a vertical list of other services is shown: All AWS Services, Compute, Storage & Content Delivery, RDS (which is also highlighted in blue), Networking, Administration & Security, Analytics, and Application Services. A large, semi-transparent watermark reading 'DO NOT DISIBUTE' is overlaid across the entire table.</p> <p>RDS</p> <p>Amazon Relational Database Service (RDS) provides familiar SQL databases while automatically managing administrative tasks.</p> <p>DynamoDB</p> <p>Amazon DynamoDB is a scalable NoSQL data store that manages distributed replicas of your data for high availability.</p>

2.1.2	<p>Click the Create Table button.</p> 
2.1.3	<p>Name the table Bootcamp2015-BLEDetections</p>
2.1.4	<p>Select a Hash primary key type, name it UUID with type String.</p> 

2.1.5	Click Continue . On the next page, we don't need to define any indexes, so click Continue .
2.1.6	<p>On the next page, we need to specify the throughput.</p>
2.1.7	We will use five Read Capacity Units and five Write Capacity Units. Click Continue .

- 2.1.8 On the next page, we can opt in for notification if we hit the capacity of our provisioned throughput.



Enter your email address to receive notifications from Basic Alarms, or if you do not want to receive any emails, untick the **Use Basic Alarms** option.

- 2.1.9 Click **Continue**. On the next page we see a summary including the cost estimate. Click **Create** to create the table.

Create Table

PRIMARY KEY ADD INDEXES (optional) PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS (optional) SUMMARY

Review

Review the specifications for the table. Be aware that the hash key, range key, and local secondary index details can be changed after the table is created.

Table Name: Bootcamp2015-BLEDetections
Primary Key Type: Hash
Hash Key Attribute: UUID (String)

Provisioned Throughput:

Index / Table	Read Capacity Units	Write Capacity Units
---------------	---------------------	----------------------

This table will not have any indexes.

Estimated Provisioned Throughput Cost: \$2.91 / month *Taxes may apply.

Use Basic Alarms: Yes
Alarm Threshold: 80%

Alarm Notification Recipients: alariter@amazon.com

Indexes:

Index Type	Index Hash Key	Index Range Key	Index Name	Projected Attributes
------------	----------------	-----------------	------------	----------------------

Back **Create**

- 2.1.10 On the dashboard, we will see the table being created for us. This will take a few moments

Amazon DynamoDB Tables		
Filter:	Explore Table	Create Table
Name	Status	Hash Key
Bootcamp2015-BLEDetections	CREATING	UUID

Task 3: Setting up AWS Lambda to write to the DynamoDB Table

Overview

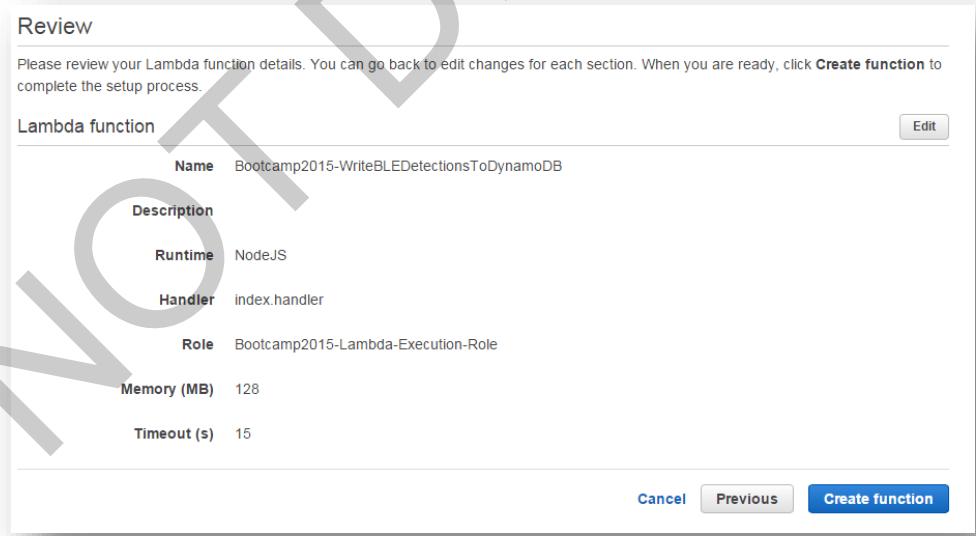
In this section, we will add a new Lambda function and bind it to the Amazon Kinesis stream. This new Lambda function will write the BLE detection data records from Amazon Kinesis to our new DynamoDB table.

Task 3-1: Create a Lambda function to write to Amazon DynamoDB

Overview

We will create a new Lambda function and use the NodeJS source code which can insert data into Amazon DynamoDB

Step	Instruction
3.1.1	<p>Browse back to the Lambda dashboard in the AWS Console. (If you already have a browser tab left open to AWS Lambda from a previous step, browse back to the main dashboard by clicking the top left Lambda link)</p>  <p>The screenshot shows the AWS Lambda Functions dashboard. At the top, there's a navigation bar with icons for AWS, Services, and Edit. Below it, the title 'Lambda > Functions' is displayed. A message says 'You have 1 Lambda function(s) using 408 bytes of code storage. Choose any Lambda function to view its details.' There are two buttons: 'Create a Lambda function' (blue) and 'Actions' (dropdown). Below these are filter and sorting options ('Function name', 'Description'). A single function entry is listed: 'Bootcamp2015-Dump-Kinesis-BLE' with the description 'An Amazon Kinesis stream processor'.</p>
3.1.2	<p>In the AWS Lambda console, click Create a Lambda function. On the 'Select blueprint' screen, press the skip button because we will be providing our own code.</p> <p>You may need to scroll down to find the Skip button.</p>
3.1.3	<p>Name the function</p> <p style="text-align: center;">Bootcamp2015-WriteBLEDetectionsToDynamoDB</p>
3.1.4	<p>In the Lambda function code section, paste the contents of Bootcamp2015-WriteBLEDetectionsToDynamoDB.lambda.js which is in the bundle you downloaded in the previous lab guides and referred to as the Desktop bundle in previous labs.</p>

3.1.5	Again, choose Bootcamp2015-Lambda-Execution-Role as the execution role for the Lambda function.																					
3.1.6	<p>In the <i>Advanced Setting</i> section, increase the Timeout to 15 seconds. We will need additional time when making other calls from this Lambda function later.</p>  <p>Advanced settings These settings allow you to control the code execution performance and costs for your function. Learn more about how Lambda pricing works.</p> <table border="1"> <tr> <td>Memory (MB)</td> <td>128</td> </tr> <tr> <td>Timeout (s)</td> <td>15</td> </tr> </table>	Memory (MB)	128	Timeout (s)	15																	
Memory (MB)	128																					
Timeout (s)	15																					
3.1.7	<p>Click Next. Review and create your function.</p>  <p>Review Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click Create function to complete the setup process.</p> <p>Lambda function</p> <table border="1"> <tr> <td>Name</td> <td>Bootcamp2015-WriteBLEDetectionsToDynamoDB</td> <td>Edit</td> </tr> <tr> <td>Description</td> <td>(empty)</td> <td></td> </tr> <tr> <td>Runtime</td> <td>NodeJS</td> <td></td> </tr> <tr> <td>Handler</td> <td>index.handler</td> <td></td> </tr> <tr> <td>Role</td> <td>Bootcamp2015-Lambda-Execution-Role</td> <td></td> </tr> <tr> <td>Memory (MB)</td> <td>128</td> <td></td> </tr> <tr> <td>Timeout (s)</td> <td>15</td> <td></td> </tr> </table> <p>Create function</p>	Name	Bootcamp2015-WriteBLEDetectionsToDynamoDB	Edit	Description	(empty)		Runtime	NodeJS		Handler	index.handler		Role	Bootcamp2015-Lambda-Execution-Role		Memory (MB)	128		Timeout (s)	15	
Name	Bootcamp2015-WriteBLEDetectionsToDynamoDB	Edit																				
Description	(empty)																					
Runtime	NodeJS																					
Handler	index.handler																					
Role	Bootcamp2015-Lambda-Execution-Role																					
Memory (MB)	128																					
Timeout (s)	15																					

- 3.1.8 Now you need to bind the Lambda function to our Amazon Kinesis stream.
Click **Event sources**.

[Lambda](#) > [Functions](#) > Bootcamp2015-WriteBLEDetectionsToDynamoDB

Use the tabs below to view your function's code, configuration, event sources, API endpoints, and metrics. You can also manually test your function with sample data. Remember that this will actually execute the code, so comment out any actions you don't want to take!

Congratulations! Your Lambda function "Bootcamp2015-WriteBLEDetectionsToDynamoDB" has been successfully created.

ARN - arn:aws:lambda:us-east-1:919696375367:function:Bootcamp2015-WriteBLEDetectionsToDynamoDB

Test **Actions** ▾

Code **Configuration** **Event sources** **API endpoints** **Monitoring**

You do not have any event sources for this function.

[Add event source](#)

- 3.1.9 Click the **Add Event Source** link.

Add event source

Configure your Lambda function to respond to events from the event sources listed below. You may also call your Lambda function directly using the AWS [mobile SDK for Android](#) and [iOS](#).

Event source type Kinesis

Kinesis stream Bootcamp2015

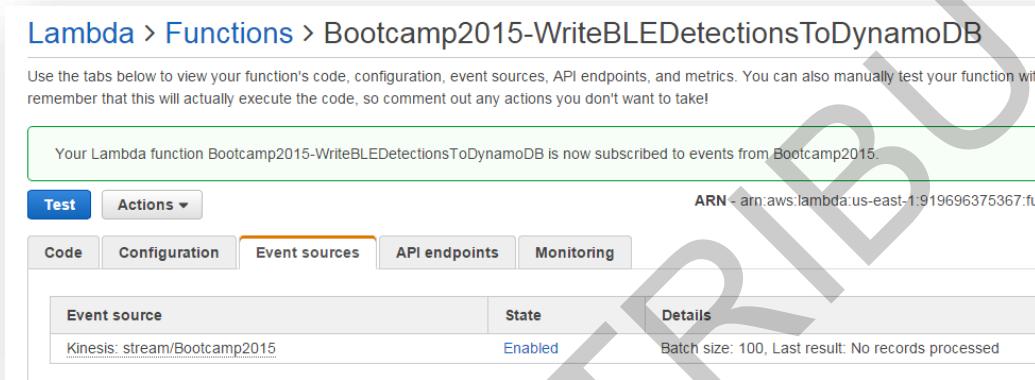
Batch size 100

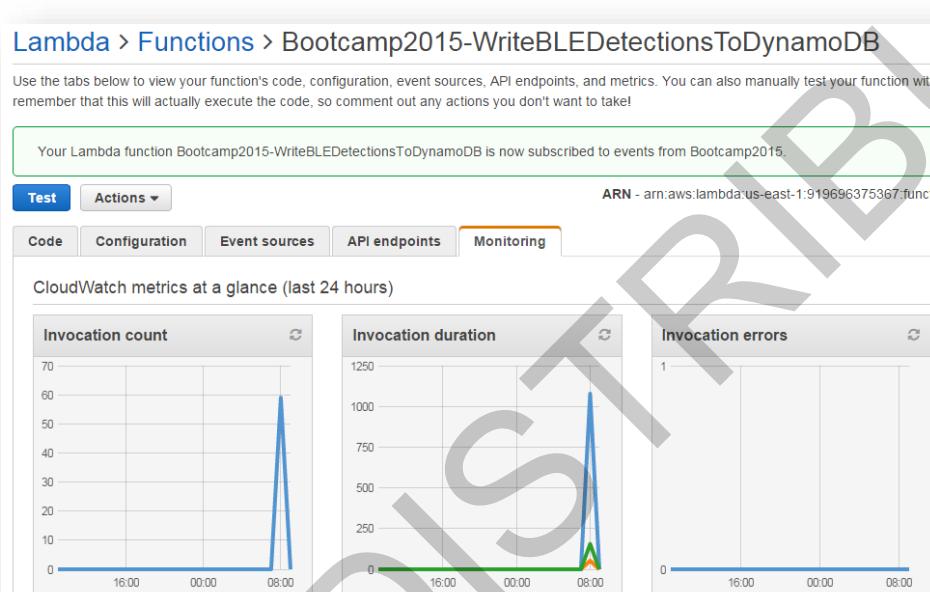
Starting position Latest

In order to read from the Kinesis stream, your execution role must have proper permissions.
[Click here to add a new policy to your execution role with Amazon Kinesis permissions.](#) (Opens in new window.)

Enable event source Enable now Enable later

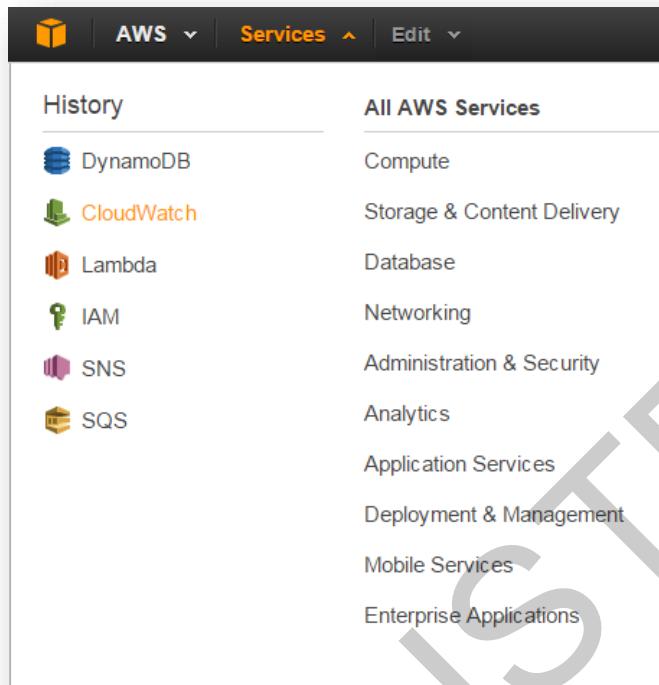
[Cancel](#) [Submit](#)

3.1.10	<p>Choose Kinesis as the event source type and then choose the Bootcamp2015 stream. Select Latest as the starting position and click the Enable Now option. Click Submit.</p> 
3.1.11	<p>We are now ready to capture Bluetooth LE advertisements on the Edison, and have them written to Amazon DynamoDB, by sending the detections to the Amazon Kinesis stream.</p>
3.1.12	<p>On your Edison, run the following:</p> <pre>cd /home/root/labs/ node test-ble-to-kinesis-smoothed-and-batched.js</pre>
3.1.13	<p>You will see the following on your SSH console</p> <pre>root@mini:~/labs# node test-ble-to-kinesis-smoothed-and-batched.js onNobleStateChange -> poweredOn Starting to Advertise this Edison... Initialising Cognito... Cognito IdentityId => us-east-1:75d8fd9f-ab70-496c-86e6-00779be52aca Starting to listen for other Edisons... New Edison discovered: 001122335566 @ -67 192.168.200.22:1(001122335566) :: 0 dBm >> 0 metres away 192.168.200.22:1(001122335566) :: -67.6590 dBm >> 1.6399 metres away 192.168.200.22:1(001122335566) :: -64.6907 dBm >> 1.3480 metres away 192.168.200.22:1(001122335566) :: -66.9693 dBm >> 1.5650 metres away 192.168.200.22:1(001122335566) :: -69.6355 dBm >> 1.8813 metres away 192.168.200.22:1(001122335566) :: -68.1884 dBm >> 1.7005 metres away 192.168.200.22:1(001122335566) :: -66.6911 dBm >> 1.5360 metres away 192.168.200.22:1(001122335566) :: -68.2646 dBm >> 1.7095 metres away</pre>

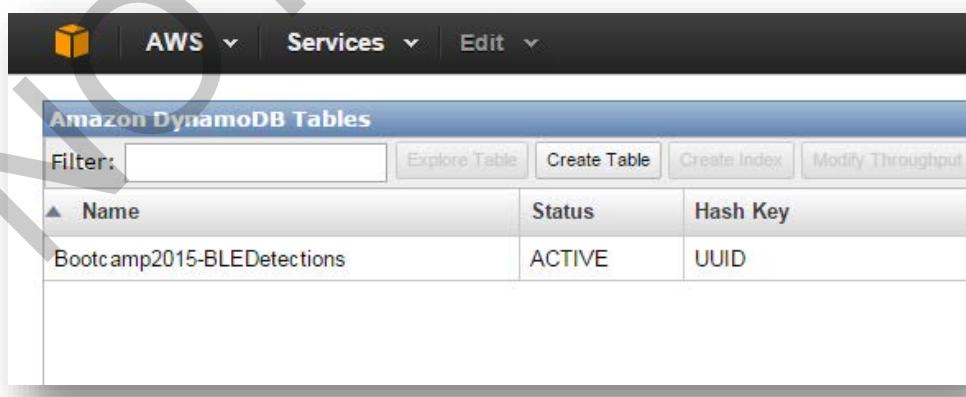
3.1.14	Note: You will only see other Edisons in range if they are using the same UUID as you are, as defined in your /etc/bootcamp-config.js file.
3.1.15	<p>Back on the AWS Lambda console, click the Monitoring tab. It may take a few moments to update the monitoring.</p>  <p>You should see some activity on the monitoring graphs as the Lambda function is executed as a result of the BLE payloads being sent to Amazon Kinesis from the Intel Edison.</p>

3.1.16	<p>Click the View logs in Cloudwatch button:</p> <pre> ▼ 2015-07-12T08:41:21.626Z c595c333-2871-11e5-a034-216d5ef1154c ===== { "Records": [{ "kinesis": { "kinesisSchemaVersion": "1.0", "partitionKey": "us-east-1:3ed528a9-570a-4097-8a88-011879abd1c0", "sequenceNumber": "4955247846464813437762292620201052008836793462541680642", "data": "eyJjbGhckV4aXN0aW5nIjp0Vmz7Sw1YXjYXIE7XR1Y3RpB25tJpbev1uM1Tjo1MTkyLjE2OCAyMDAuNzoxTiwiidXVpZC16TjAwHTEyMjMzNTU2N1sInJzc2k10i0MyiV29uZmlkZm5jZSLGM2YUm2csImZpDH1cnKU1nTSS1G110MS42NDg2Li1v2fsv3vSYR1ZEKpc3Rhbmhlij01Mc4yNQzLiwib6FzdeRldGvjG1vb1RpbwVzG0tcC16Tj1wMTUDctMTJU0Dg6IDE6TKu1jEWL1TmFycmFSGV07W0a0u9ucyT6hmVshtMdfQ==" }, "eventSource": "aws:kinesis", "eventVersion": "1.0", "eventId": "shardId-000000000004:49552478464648134437762292620201052008836793462541680642", "eventName": "#\$aws:kinesis:record", "invokeIdentityArn": "arn:aws:iam::919696375367:role/Bootcamp2015-Lambda-Execution-Role", "awsRegion": "us-east-1", "eventSourceARN": "arn:aws:kinesis:us-east-1:919696375367:stream/Bootcamp2015" }] } ▼ 2015-07-12T08:41:21.626Z c595c333-2871-11e5-a034-216d5ef1154c { name: '192.168.200.7:1', uid: '001122335566', rssi: -53, confidence: 36.37, filteredRSSI: '-51.6486', calculatedDistance: '0.2643', lastDetectionTimestamp: '2015-07-12T08:41:19.619Z', arrayDetections: null } ▼ 2015-07-12T08:41:21.626Z c595c333-2871-11e5-a034-216d5ef1154c Detection for UUID: 001122335566 (192.168.200.7:1) ▼ 2015-07-12T08:41:21.626Z c595c333-2871-11e5-a034-216d5ef1154c Total of 1 detections found ▼ 2015-07-12T08:41:21.626Z c595c333-2871-11e5-a034-216d5ef1154c Updating DynamoDB record for 001122335566 ▼ 2015-07-12T08:41:21.708Z c595c333-2871-11e5-a034-216d5ef1154c DynamoDB callback for 001122335566 OK. 0 callback(s) to go... END RequestId: c595c333-2871-11e5-a034-216d5ef1154c Duration: 99.51 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 43 MB ▼ REPORT RequestId: c595c333-2871-11e5-a034-216d5ef1154c Duration: 99.51 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 43 MB </pre>
3.1.17	<p>The above shows an example output. Note that the update to Amazon DynamoDB is successful.</p> <pre> Detection for UUID: 001122335566 (192.168.200.22:1) Total of 1 detections found Updating DynamoDB record for 001122335566 DynamoDB callback for 001122335566 OK. 0 callback(s) to go... </pre>

- 3.1.18 In the AWS console, browse to the Amazon DynamoDB dashboard:



- 3.1.19 Click the **Bootcamp2015-BLEDetections** table in the list, and select **Explore Table**.



3.1.20	<p>You will see the detected Edison devices in the DynamoDB table if all is working as expected.</p> 
3.1.21	<p>If you click the Go button a few times, you should see the data updating in real-time.</p>
3.1.22	<p>Press 'Ctrl-c' to stop the test-ble-to-kinesis-smoothed-and-batched.js code from running and move onto the next section.</p>

Lab 6: Messaging with Amazon SQS, Amazon SNS and the MQTT protocol

Overview	<p>In this lab, we introduce the Amazon Simple Queue Service (SQS), the Amazon Simple Notification Service (SNS) and the MQ Telemetry Transfer protocol (MQTT). We will then send and receive messages to/from the Edison using the various messaging techniques.</p> <p>To successfully complete this module, you will need to work in pairs so you can establish an MQTT connection between two Edison devices.</p> <p>We will refer to one of the Edisons as the Primary Edison, and the other as the Secondary Edison. In your pair, decide which of you will be primary and secondary. If you have time, you can swap roles over so you both observe the behavior.</p> <p>The Primary is the device that sends a message to the secondary's SQS queue asking it to pair with the Primary appliance. As we progress further through the Bootcamp, the SQS message requesting the pairing will be initiated by the end user via the companion mobile app, but for the moment, we will send the SQS message from the primary Edison.</p>
-----------------	--

Objectives	<p>In this lab, you will:</p> <ul style="list-style-type: none">• Create an SQS queue using the console• Send messages to an SQS queue using NodeJS• Receive messages from an SQS queue using NodeJS• Create an SNS topic• Subscribe a Lambda function to an SNS topic• Create a Lambda function that logs SNS events received.
-------------------	--

Pre-requisites	This lab requires: <ul style="list-style-type: none">Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">Note: The <i>qwikLABS</i> lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.For Microsoft Windows users: Administrator access to the computer.An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).
Duration	30 minutes

Task 1: Creating an SQS Queue

Overview	In this section we will create a new SQS queue from the AWS Management Console. Amazon Simple Queue Service (SQS) is a fast, reliable, scalable, fully managed message queuing service. SQS makes it simple and cost-effective to decouple the components of a cloud application. You can use SQS to transmit any volume of data, at any level of throughput, without losing messages or requiring other services to be always available.
-----------------	---

Task 1-1: Create an SQS Queue

Overview	In this task, we will create the SQS queue.
-----------------	---

Step	Instruction
------	-------------

1.1.1

Login to the AWS Management Console from your Qwiklabs Environment and choose **Services - Application Services – SQS**.

All AWS Services

Compute

Storage & Content Delivery

Database

Networking

Administration & Security

Analytics

Application Services

Deployment & Management

Mobile Services

Enterprise Applications

SQS

SWF

Amazon Simple Queue Service (SQS) offers a reliable, highly scalable, hosted queue for storing messages.

Amazon Simple Workflow (SWF) coordinates all of the processing steps within an application.

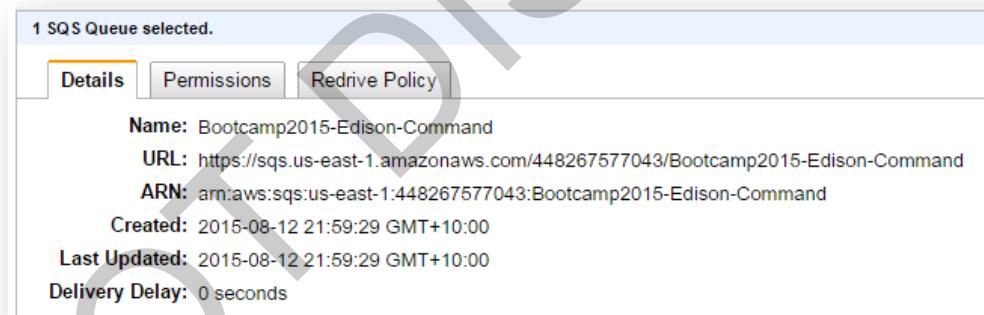
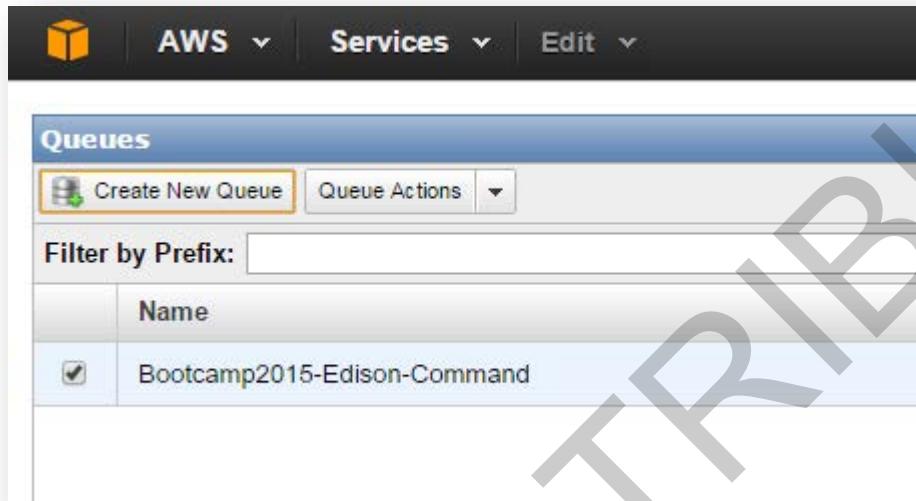
1.1.2

Click the **Create New Queue** button in the top section.



1.1.3	<p>In the window that appears, name the queue Bootcamp2015-Edison-Command and leave all the other values as defaults.</p> <p>Click Create Queue in the bottom right of the window to create your queue.</p> <pre> <table border="1"> <thead> <tr> <th colspan="2">Create New Queue</th> </tr> </thead> <tbody> <tr> <td colspan="2"> Please enter a name for your new queue. Queue names must be 1-80 characters in length and be composed of alphanumeric characters, hyphens (-), and underscores (_). Your queue will be created in the US East (N. Virginia) region. </td> </tr> <tr> <td>Region:</td> <td>US East (N. Virginia)</td> </tr> <tr> <td>Queue Name:</td> <td>Bootcamp2015-Edison-Command</td> </tr> <tr> <td colspan="2"> Configure your new queue by setting queue attributes (optional). </td> </tr> <tr> <td colspan="2"> Queue Settings </td> </tr> <tr> <td>Default Visibility Timeout:</td> <td>30 <input type="button" value="seconds"/> Value must be between 0 seconds and 12 hours.</td> </tr> <tr> <td>Message Retention Period:</td> <td>4 <input type="button" value="days"/> Value must be between 1 minute and 14 days.</td> </tr> <tr> <td>Maximum Message Size:</td> <td>256 <input type="button" value="KB"/> Value must be between 1 and 256 KB.</td> </tr> <tr> <td>Delivery Delay:</td> <td>0 <input type="button" value="seconds"/> Value must be between 0 seconds and 15 minutes.</td> </tr> <tr> <td>Receive Message Wait Time:</td> <td>0 <input type="button" value="seconds"/> Value must be between 0 and 20 seconds.</td> </tr> </tbody> </table> </pre>	Create New Queue		Please enter a name for your new queue. Queue names must be 1-80 characters in length and be composed of alphanumeric characters, hyphens (-), and underscores (_). Your queue will be created in the US East (N. Virginia) region.		Region:	US East (N. Virginia)	Queue Name:	Bootcamp2015-Edison-Command	Configure your new queue by setting queue attributes (optional).		Queue Settings		Default Visibility Timeout:	30 <input type="button" value="seconds"/> Value must be between 0 seconds and 12 hours.	Message Retention Period:	4 <input type="button" value="days"/> Value must be between 1 minute and 14 days.	Maximum Message Size:	256 <input type="button" value="KB"/> Value must be between 1 and 256 KB.	Delivery Delay:	0 <input type="button" value="seconds"/> Value must be between 0 seconds and 15 minutes.	Receive Message Wait Time:	0 <input type="button" value="seconds"/> Value must be between 0 and 20 seconds.
Create New Queue																							
Please enter a name for your new queue. Queue names must be 1-80 characters in length and be composed of alphanumeric characters, hyphens (-), and underscores (_). Your queue will be created in the US East (N. Virginia) region.																							
Region:	US East (N. Virginia)																						
Queue Name:	Bootcamp2015-Edison-Command																						
Configure your new queue by setting queue attributes (optional).																							
Queue Settings																							
Default Visibility Timeout:	30 <input type="button" value="seconds"/> Value must be between 0 seconds and 12 hours.																						
Message Retention Period:	4 <input type="button" value="days"/> Value must be between 1 minute and 14 days.																						
Maximum Message Size:	256 <input type="button" value="KB"/> Value must be between 1 and 256 KB.																						
Delivery Delay:	0 <input type="button" value="seconds"/> Value must be between 0 seconds and 15 minutes.																						
Receive Message Wait Time:	0 <input type="button" value="seconds"/> Value must be between 0 and 20 seconds.																						

- 1.1.4 The queue will be created. From the dashboard, select the new queue you just created, and view the details in the lower panel.



Notice how the URL of the queue is constructed using your Account number and the name of the queue. The ARN, Amazon Resource Name, is also a well-defined pattern made up of the account number and queue name. Because of this, these details can be derived and used to configure the relevant IAM roles and bootcamp-config.js file ahead of time. If you are following the naming convention specified in the Lab guides, you are already correctly configured to allow the Edison to receive messages from the new queue.

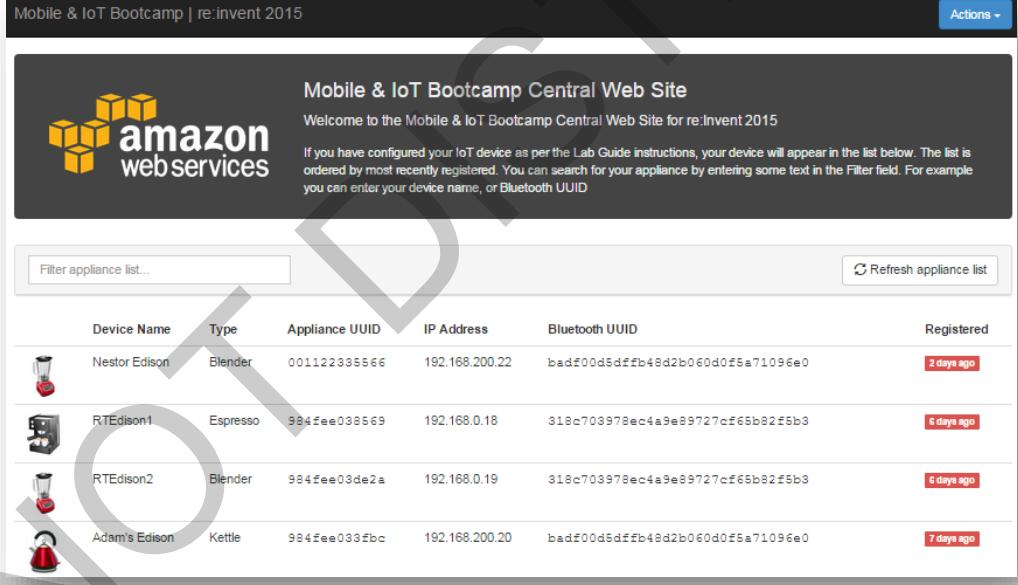
Task 2: Sending a Message from the Console and receiving it on the Edison

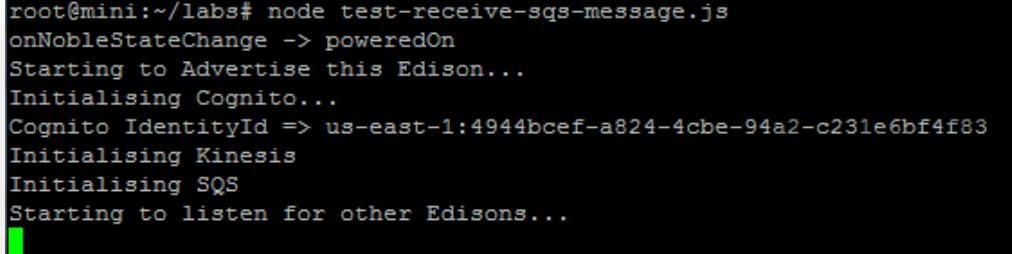
Overview	In this section we will use the SQS queue we just created and send a message to the Edison, both manually and programmatically.
-----------------	---

Task 2-1: Send an SQS Message manually

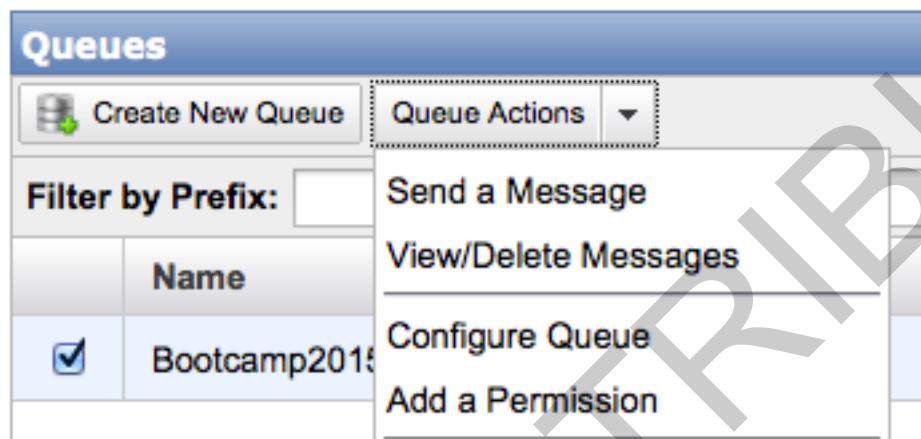
Overview	In this section, we will send a manual SQS message to your Edison.
-----------------	---

Step	Instruction

2.1.1	<p>Connect to your Edison device using the SSH protocol, if you are not already still connected.</p> <p>Windows users</p> <p>Use the Putty application to connect to your Edison on the IP address shown in the Central web site for your device.</p> <p>Mac users</p> <p>Use the terminal to SSH to your Edison on the IP address shown in the Central web site for your device.</p> <p>You can browse to the Central web site and determine the IP address for your device. Go to the following location in your browser:</p> <p>http://bit.ly/mobile-iot-bootcamp</p>  <p>The screenshot shows a web page titled "Mobile & IoT Bootcamp Central Web Site". The page features the Amazon Web Services logo. It displays a table of registered devices with the following data:</p> <table border="1"> <thead> <tr> <th>Device Name</th> <th>Type</th> <th>Appliance UUID</th> <th>IP Address</th> <th>Bluetooth UUID</th> <th>Registered</th> </tr> </thead> <tbody> <tr> <td>Nestor Edison</td> <td>Blender</td> <td>001122335566</td> <td>192.168.200.22</td> <td>badf00d5dfffb48d2b060d0f5a71096e0</td> <td>2 days ago</td> </tr> <tr> <td>RTEdison1</td> <td>Espresso</td> <td>984f0ee038569</td> <td>192.168.0.18</td> <td>318c703978ec4a9e89727cf65b82f5b3</td> <td>6 days ago</td> </tr> <tr> <td>RTEdison2</td> <td>Blender</td> <td>984f0ee03de2a</td> <td>192.168.0.19</td> <td>318c703978ec4a9e89727cf65b82f5b3</td> <td>6 days ago</td> </tr> <tr> <td>Adam's Edison</td> <td>Kettle</td> <td>984f0ee033f0c</td> <td>192.168.200.20</td> <td>badf00d5dfffb48d2b060d0f5a71096e0</td> <td>7 days ago</td> </tr> </tbody> </table> <p>Locate your device by Name, and determine your IP address. You can use the Filter feature to enter in the name of your device to help locate your Edison in the list.</p>	Device Name	Type	Appliance UUID	IP Address	Bluetooth UUID	Registered	Nestor Edison	Blender	001122335566	192.168.200.22	badf00d5dfffb48d2b060d0f5a71096e0	2 days ago	RTEdison1	Espresso	984f0ee038569	192.168.0.18	318c703978ec4a9e89727cf65b82f5b3	6 days ago	RTEdison2	Blender	984f0ee03de2a	192.168.0.19	318c703978ec4a9e89727cf65b82f5b3	6 days ago	Adam's Edison	Kettle	984f0ee033f0c	192.168.200.20	badf00d5dfffb48d2b060d0f5a71096e0	7 days ago
Device Name	Type	Appliance UUID	IP Address	Bluetooth UUID	Registered																										
Nestor Edison	Blender	001122335566	192.168.200.22	badf00d5dfffb48d2b060d0f5a71096e0	2 days ago																										
RTEdison1	Espresso	984f0ee038569	192.168.0.18	318c703978ec4a9e89727cf65b82f5b3	6 days ago																										
RTEdison2	Blender	984f0ee03de2a	192.168.0.19	318c703978ec4a9e89727cf65b82f5b3	6 days ago																										
Adam's Edison	Kettle	984f0ee033f0c	192.168.200.20	badf00d5dfffb48d2b060d0f5a71096e0	7 days ago																										

2.1.2	<p>To test receiving messages via SQS, run the test file with:</p> <pre>cd /home/root/labs node test-receive-sqs-message.js</pre> 
2.1.3	<p>The script will start, initialize with Cognito, set up to send BLE detection events to the Kinesis stream and then setup a polling loop against the SQS queue URL you set in the configuration file. It will then pause, waiting for a message to arrive.</p> <p>If there is another Edison appliance configured to use the same UUID for BLE as your Edison, you will see detection data appear in the console window, and the data will be sent to Kinesis, and processed by the Lambda function we set up in the previous Lab.</p> <pre>root@mini:~/labs# node test-receive-sqs-message.js onNobleStateChange -> poweredOn Starting to Advertise this Edison... Initialising Cognito... Cognito IdentityId => us-east-1:4944bcef-a824-4cbe-94a2-c231e6bf4f83 Initialising Kinesis Initialising SQS Starting to listen for other Edisons... New Edison discovered: 001122335566 @ -75 192.168.200.22:1(001122335566) :: 0 dBm >> 0 metres away 192.168.200.22:1(001122335566) :: -67.5515 dBm >> 1.6279 metres away 192.168.200.22:1(001122335566) :: -68.1848 dBm >> 1.7001 metres away 192.168.200.22:1(001122335566) :: -66.5414 dBm >> 1.5208 metres away 192.168.200.22:1(001122335566) :: -63.8210 dBm >> 1.2764 metres away 192.168.200.22:1(001122335566) :: -65.4403 dBm >> 1.4145 metres away</pre> <p>We will send a message to the Edison in the next steps.</p>

- 2.1.4 Browse to the **SQS** dashboard in the AWS console. Select the queue you created earlier, and from the **Queue Actions** menu, select **Send a Message**:

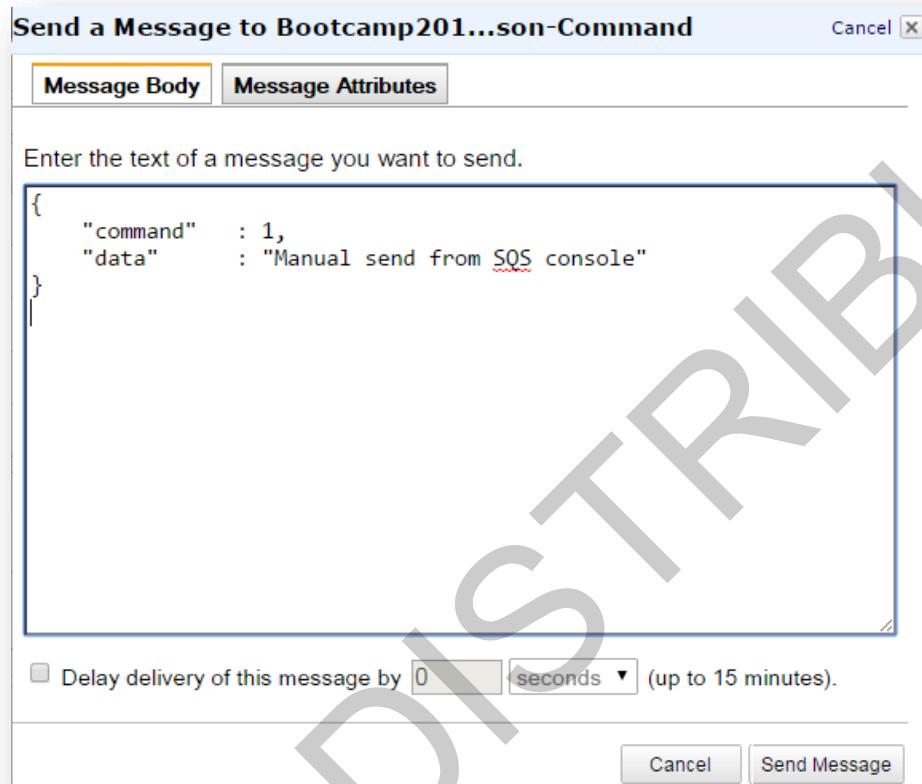


- 2.1.5 Enter the following JSON text in the dialog box.

```
{  
    "command" : 1,  
    "data" : "Manual send from SQS console"  
}
```

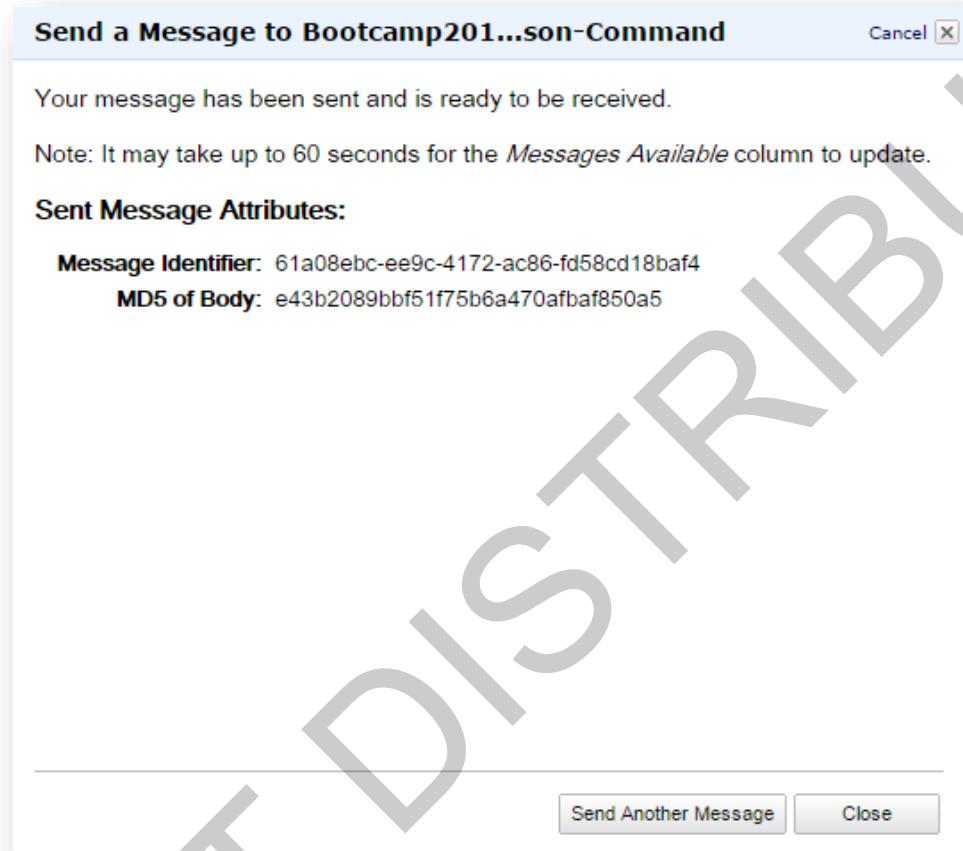
2.1.6

Click **Send Message**



2.1.7

The console will show the following confirmation message:



2.1.8

On the Edison, you will see the following:

```
root@mini:~/labs# node test-receive-sqs-message.js
onNobleStateChange -> poweredOn
Starting to Advertise this Edison...
Initialising Cognito...
Cognito IdentityId => us-east-1:4944bcef-a824-4cbe-94a2-c231e6bf4f83
Initialising Kinesis
Initialising SQS
Starting to listen for other Edisons...
*** Test message received! -> Manual send from SQS console
```

2.1.9	The message will have been successfully received if you see the output shown above. In the next section, we will send the message to the SQS queue programmatically.
-------	--

Task 2-2: Send a test message programmatically, via NodeJS

Overview	In this section, we will send a test message using NodeJS. The NodeJS program that sends the message will authenticate anonymously and gain access to cloud resources using Cognito .
----------	--

Step	Instruction
2.2.1	<p>SSH into your Edison in another session so that the previous script (<code>test-receive-sqs-message.js</code>) is still running. For example on Mac you would open a new terminal window and type ‘ssh root@ipaddress’.</p> <p>Run another test program in the new SSH session as follows:</p> <pre>cd /home/root/labs node test-send-sqs-message.js</pre> <div style="background-color: black; color: white; padding: 10px;"> <pre>Using username "root". root@192.168.200.18's password: root@mini:~# cd labs/ root@mini:~/labs# node test-send-sqs-message.js Initialising Cognito... Cognito IdentityId => us-east-1:8abb81af-b365-45f7-8367-8bb21a13836b Sending message to SQS queue... sent SQS MessageId: d1f4ce4b-3a7d-4f36-b4bf-018109615f4a root@mini:~/labs#</pre> </div>

2.2.2	<p>Now, in the other SSH session window (assuming you still have <code>test-receive-sqs-message.js</code> still running – if not, run it now) you should see this:</p> <pre>root@mini:~/labs# node test-receive-sqs-message.js onNobleStateChange -> poweredOn Starting to Advertise this Edison... Initialising Cognito... Cognito IdentityId => us-east-1:4944bcef-a824-4cbe-94a2-c231e6bf4f83 Initialising Kinesis Initialising SQS Starting to listen for other Edisons... *** Test message received! -> Manual send from SQS console *** Test message received! -> Hello, World!</pre> <p><i>Remember, if your partner is running their Edison tests at the same time, you will see detection data being sent to Kinesis as well.</i></p>
2.2.3	This confirms that your Edison is correctly receiving messages via SQS.
2.2.4	Press ‘Ctrl-c’ to stop the Node.js code from running in both terminal windows and move onto the next section (you can close one of the connections at this point).

Task 3: Update SQS Queue permissions to allow your partner's Edison to send to your Queue

Overview	<p>The SQS queue we created in an earlier task is secure by default. Only you, as the owner of the queue, can send messages or receive message or otherwise interact with the queue.</p> <p>It is useful sometimes to allow another user principal to access an SQS queue you own. For example, you may want to allow third-parties to send messages to an SQS queue as a way of decoupling</p>
-----------------	---

	<p>workloads between AWS accounts.</p> <p>For this task, we will set up the queue and run some tests to demonstrate how we can allow another AWS account to send messages to our queue, and have the queue message received on the Edison.</p>
--	--

Task 3-1: Grant permissions to your Partner's Account to send to your SQS Queue

Overview	In this section we will update the SQS queue that we created in the previous tasks to allow your partner's Edison to send messages to your Edison's SQS queue.
-----------------	--

Step	Instruction
3.1.1	Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Application Services – SQS .

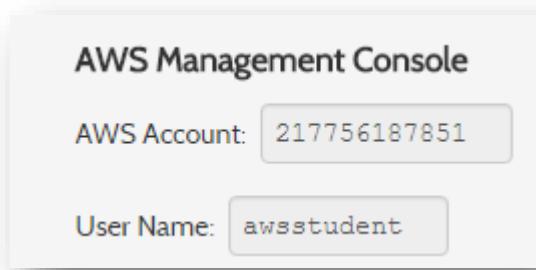
- 3.1.2 In the SQS dashboard of the AWS Console, select your queue and in the bottom pane, select the **Permissions** tab.

The screenshot shows the AWS SQS Queue Details page. At the top, there's a 'Queues' header with 'Create New Queue' and 'Queue Actions' buttons. Below that is a 'Filter by Prefix:' input field. The main area displays a table with one row, 'Bootcamp2015-Edison-Command', which has a checked checkbox next to it. A message below the table says '1 SQS Queue selected.' Below the table, there are three tabs: 'Details' (disabled), 'Permissions' (selected and highlighted in orange), and 'Redrive Policy'. Under the 'Permissions' tab, there's a table with columns 'Effect', 'Principals', and 'Actions'. A note states: 'This queue has an empty [SQS Queue Access Policy](#). This means that only the queue owner is allowed to access the queue.' At the bottom of this section are three buttons: 'Add a Permission' (with a plus sign icon), 'Edit Policy Document (Advanced)' (with a pencil icon), and 'What's an SQS Queue Access Policy?' (with a question mark icon).

- 3.1.3 Click the **Add a Permission** button on the left.

The screenshot shows the 'Add a Permission to Bootcamp2015-Edison-Command' dialog box. At the top, it says 'Add a Permission to Bootcamp2015-Edison-Command'. Below that, a note says 'Permissions enable you to control which operations a user can perform on a queue. [Learn more about access control concepts.](#)' Underneath, there are settings for 'Effect': 'Allow' (radio button selected) and 'Deny'. There's also a checkbox for 'Everybody (*)'. The 'Principal' field contains 'aws account number(s)' with a note: 'Use commas between multiple values.' The 'Actions' field has a dropdown menu set to '--- No Specific Actions ---' and a checkbox for 'All SQS Actions (SQS:*)'. At the bottom is a blue 'Add Conditions (optional)' link.

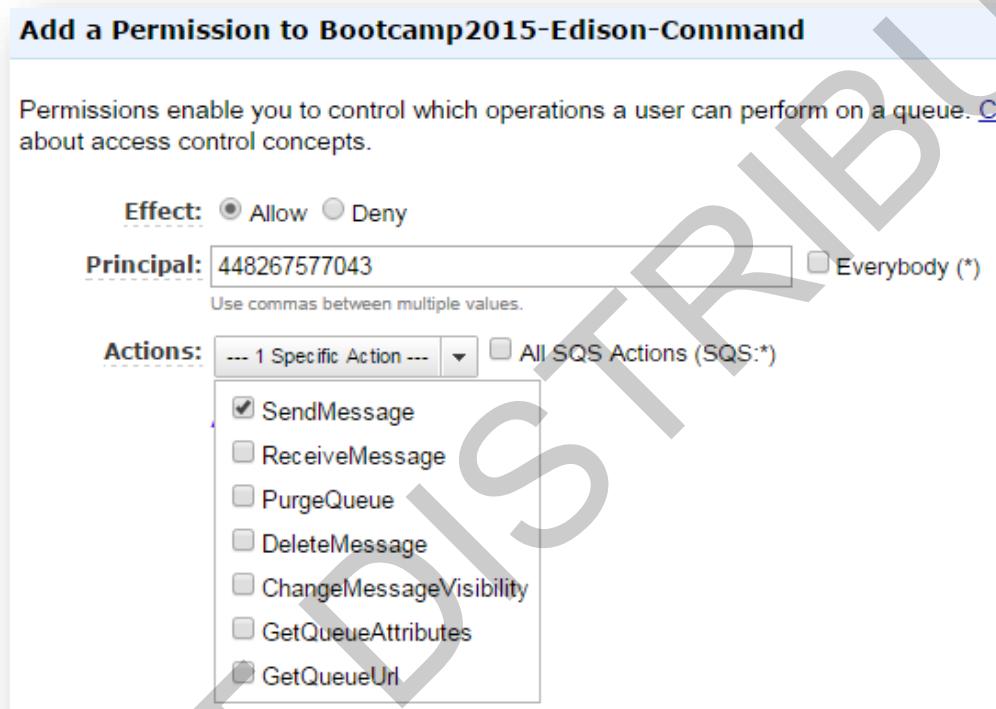
- 3.1.4 When the ‘**Add a permission**’ panel appears, ask your partner what their account number is, and type it into the **Principal** field.
- Your partner can take their account number from the Qwiklabs dashboard, where they started the Qwiklab from. An example of the Account Id is shown below:



Note: Your account Id will be different! You cannot use the account Id shown above, you must use the account Id for the Qwiklabs account you are using for the Bootcamp!

A screenshot of the "Add a Permission to Bootcamp2015-Edison-Command" dialog box. The title bar says "Add a Permission to Bootcamp2015-Edison-Command". The main area contains text about permissions and a link to learn more about access control concepts. There are two radio buttons for "Effect": "Allow" (selected) and "Deny". A text input field for "Principal" contains the value "448267577043". To the right of this input field is a checkbox for "Everybody (*)". Below the principal input field is a note: "Use commas between multiple values.". A dropdown menu for "Actions" is open, showing "No Specific Actions" and "All SQS Actions (SQS:*)". At the bottom is a button for "Add Conditions (optional)".

- 3.1.5 In the **Actions** field, select **SendMessage** to allow your partner to send a message to your queue. Both the Primary and Secondary Edison users should complete this step, so that you can try switching roles in this lab if there is time.



- 3.1.6 Click **Add Permission**.

A screenshot of a dialog box with two buttons: 'Cancel' and 'Add Permission'.

Task 4: Updating a test script on the ‘Primary’ Edison, to send Messages to the SQS Queue of the ‘Secondary’ Edison

Overview	In this section we will use a script on the Primary Edison, edit it to add the correct details for the SQS queue of the Secondary Edison, and then run it. This will cause an SQS message to be sent from the Primary Edison to the SQS queue bound to the Secondary Edison
-----------------	---

Task 4-1: Update Script and Test

Overview	In this section we will update the script and test sending messages to a secondary Edison
-----------------	---

Step	Instruction
4.1.1	<p>Edit the file <code>primary-send-sqs-message-to-initiate-pair.js</code> and locate the configuration section:</p> <pre>cd /home/root/labs vi primary-send-sqs-message-to-initiate-pair.js</pre> <p>Note that, like you did before, in order to enter the “edit mode” with vi, you must press the i key on your keyboard.</p>

4.1.2	<pre>// // CONFIGURATION: Add the Url to your partner's SQS Queue here! // var SQS_URL_QUEUE_TO_SECONDARY_EDISON = "https://sq.us-east-1.amazonaws.com/XXXXXX</pre> <p>Note the 'XXXX' in the definition of SQS_URL_QUEUE_TO_SECONDARY_EDISON? This needs to be updated to reference your partner's SQS Queue URL.</p> <p>If you have been following the recommended naming convention in the labs, you only need to replace the XXXXXX with your partner's account Id.</p> <p>This is the same account Id you previously entered into the Permissions tab of your SQS queue (refer 3.1.4 above)</p>
4.1.3	<p>If you are using vi to edit this file you can save it by pressing 'Esc' (escape key) followed by</p> <p>:x</p> <p>and pressing Enter (don't forget the colon ':' before the 'x').</p> <p>After you edit the definition, save the file on the Edison.</p>
4.1.4	<p>Now that the permissions are set up, on the primary Edison, run the following:</p> <pre>cd /home/root/labs node primary-send-sqs-message-to-initiate-pair.js</pre> <div style="background-color: black; color: green; padding: 10px;"> <pre>root@mini:~# node 8-primary-send-sqs-message-to-initiate-pair.js Initialising Cognito... Cognito IdentityId => us-east-1:afd33c8a-e7d9-40b0-916b-ff75f5a8147a SQS message sent ok</pre> </div>

4.1.5	<p>The program will connect to Cognito, display the Cognito Id (just for debug) and then send the message to the SQS queue.</p> <p>On the secondary Edison, run the following:</p> <pre>cd /home/root/labs node secondary-pair-using-mqtt.js</pre>
4.1.6	<p>The secondary Edison will connect to Cognito as well (but a different account and Cognito pool), begin advertising via BLE and also start listening for other Edisons. We don't need the BLE functionality in this example, but because we are building up the complete final project, we will leave this code in the program and just ignore the functionality for now.</p> <pre>root@mini2:~# node 8-secondary-pair-using-mqtt.js onNobleStateChange -> poweredOn Starting to Advertise this Edison... Initialising Cognito... Cognito IdentityId => us-east-1:a79082dc-6ff1-4810-aea1-9adc2f89ef50 Starting to listen for other Edisons...</pre>
4.1.7	<p>When the SQS message has been sent from the primary, and is received by the secondary, you will see the following on the secondary:</p> <pre>*** COMMAND_PAIR_REQUEST received from SQS -> Pair with 192.168.200.18 onMQTTClientConnect() Now subscribed to other Edison. In a couple of seconds, I will ask the other Edison a question...</pre>

4.1.8	<p>This indicates that the secondary has received the SQS message which is a request to Pair with the primary Edison, identified by its IP address. It also shows a message indicating that it will wait a few seconds and will then send a query directly to the other Edison via the now-established MQTT connection.</p> <p>When time is up, the MQTT message will be sent. The message that is sent will look like this:</p> <pre>{ "command" : 1, "data" : "100", "sender" : "192.168.200.21" }</pre> <p>In this case, the command value of 1 means <code>COMMAND_QUERY</code> – a request to have the receiving Edison (in this case, the Primary) send back a standard response to some pre-arranged query.</p> <p>In our case, it will respond by sending back a message that contains the data element that the secondary sent to it in the <code>COMMAND_QUERY</code> object, just to prove the data is being transmitted in both directions.</p>
4.1.9	<p>On the primary Edison, you will see the program receive the MQTT message:</p> <pre>COMMAND_QUERY from 192.168.200.7 The other Edison passed the data '100' I will respond to this by sending a message back containing the same data... MQTT message has been sent</pre>
4.1.10	<p>The text above shows how the primary Edison received the MQTT message containing the <code>COMMAND_QUERY</code> command. It responded by sending back a message containing the same data (in this test case, it was '100' that was sent in) that was sent by the secondary.</p>

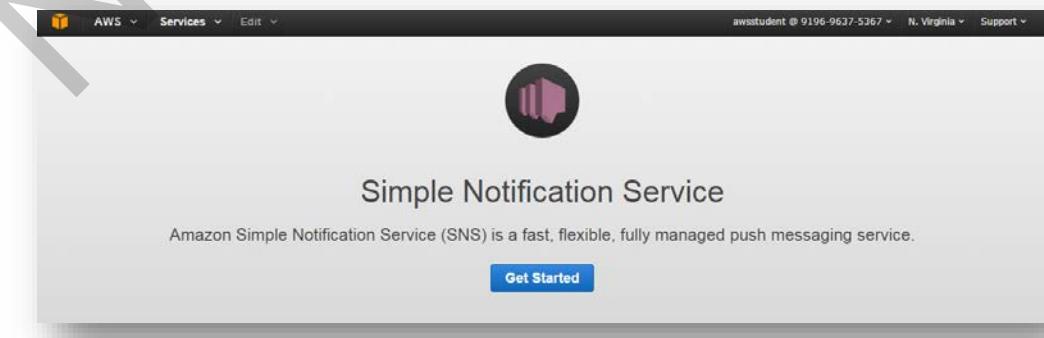
4.1.11	<p>On receiving this, the secondary will display the following:</p> <pre>COMMAND_RESPONSE from 192.168.200.18 "You said 100 - here is my response ;)"</pre>
4.1.12	<p>The whole interaction looks like this, with the Primary Edison on the right, and the Secondary on the left:</p> <pre>root@mini1:~# node s-primary-send-sqs-message-to-initiate-pair.js Initialising Cognito... Cognito IdentityId => us-east-1:afdf3c8a-e7d9-40b0-916b-ff75f5a8147a SQS message sent to COMMAND_QUERY from 192.168.200.7 The other Edison passed the data '100' I will respond to this by sending a message back containing the same data... MQTT message has been sent 0 root@mini2:~# node s-secondary-pair-using-mqtt.js onNoblesStateChange -> powerdown Starting to Advertise this Edison... Initialising Cognito... Cognito IdentityId => us-east-1:79082dc-6ff1-4810-aec1-9adc2f89ef50 Starting to listen for other Edisons... *** COMMAND_PAIR_REQUEST received from SQS -> Pair with 192.168.200.18 onMQTClientConnect() Now subscribed to other Edisons. In a couple of seconds, I will ask the other Edison a question... COMMAND_RESPONSE from 192.168.200.18 "You said 100 - here is my response ;)"</pre> <p>The purpose of this demonstration is to show how we can use an SQS message sent from a third-party to initiate an MQTT pairing between two Edisons.</p> <p>In our system, we will use this technique to establish a bi-directional connection between two Edisons so they can act as a conduit for our 'chat' application, running on two mobile devices, later in the Bootcamp.</p>

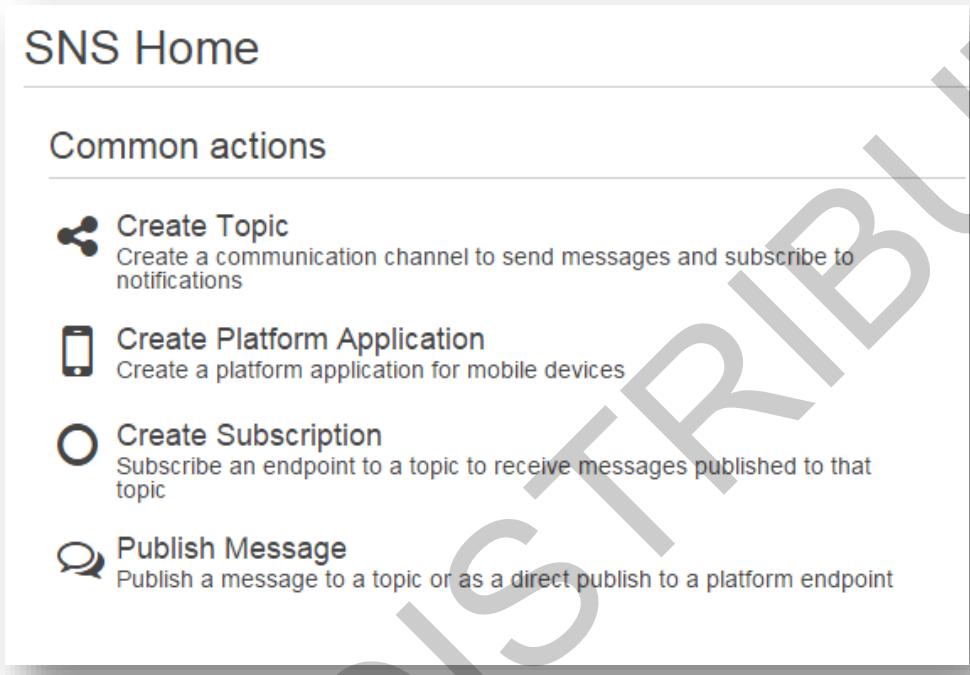
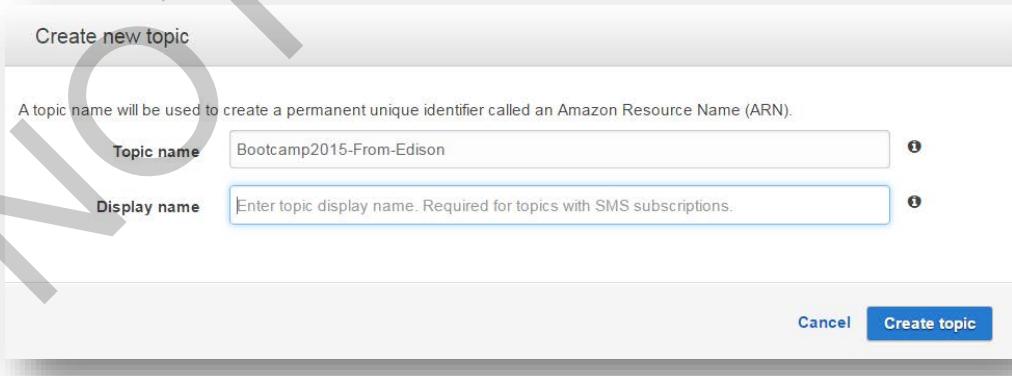
Task 5: Integrating with SNS for inbound messages from Edison

Overview	<p>In this task, we will create a new SNS topic, and send a message to it.</p>
-----------------	--

Task 5-1: Create an SNS Topic

Overview	In this lab, we will demonstrate how we can send a message to an SNS topic, but first we need to create this SNS topic
-----------------	--

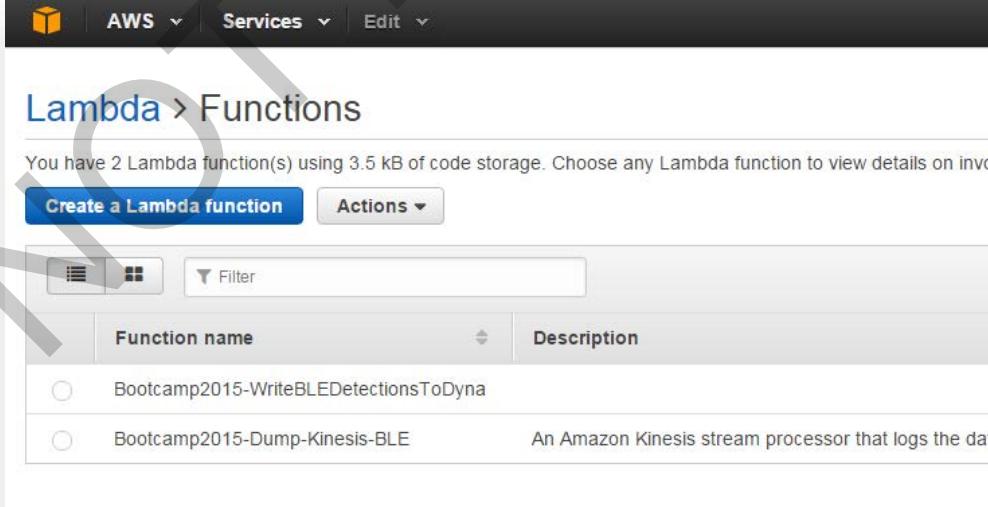
Step	Instruction
5.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Mobile Services – SNS.</p>  A screenshot of the AWS Management Console navigation bar. The 'Mobile Services' option is highlighted in blue, indicating it is selected. Other options like All AWS Services, Compute, Storage & Content Delivery, Database, Networking, Administration & Security, Analytics, Application Services, Deployment & Management, Device Farm, and Enterprise Applications are also visible. <p>5.1.2 Click the Get Started button.</p>  A screenshot of the Amazon Simple Notification Service (SNS) landing page. It features a large purple speech bubble icon and the text "Simple Notification Service". Below that, it says "Amazon Simple Notification Service (SNS) is a fast, flexible, fully managed push messaging service." A prominent blue "Get Started" button is at the bottom.

5.1.3	<p>Click Create Topic.</p> 
5.1.4	<p>Enter Bootcamp2015-From-Edison as the topic name.</p> 
5.1.5	<p>Click the Create Topic button to create the new topic.</p>

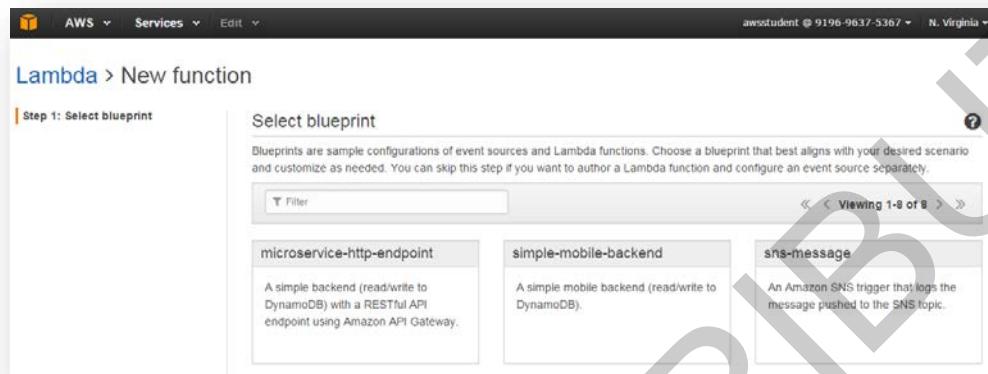
Task 5-2: Create a Lambda Function and subscribe it to the new SNS Topic

Overview

We will now create the Lambda functions that will process the messages sent to the SNS topic.

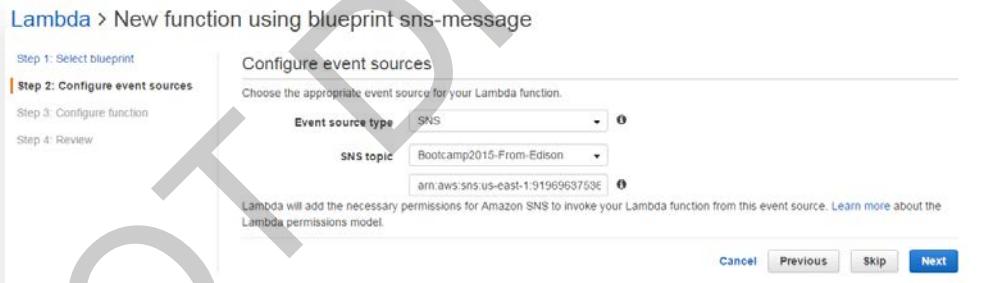
Step	Instruction
5.2.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Compute- Lambda.</p> 
5.2.2	<p>The Lambda console is displayed here.</p> 

5.2.3 Click **Create a Lambda Function**, then choose the **sns-message** blueprint:



5.2.4 Choose the SNS topic we just created (*Bootcamp2015-From-Edison*) from the drop down list. This will populate the SNS topic's ARN.

Click **Next**.



5.2.5

Name the Lambda function

Bootcamp2015-DumpMessagesFromSNS

and leave the code section as it is:

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name* Bootcamp2015-DumpMessagesFromSNS

Description An Amazon SNS trigger that logs the message pushed to the

Runtime* Node.js

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk). If your code requires custom libraries, you can upload your code and libraries as a ZIP file. [Learn more](#) about deploying Lambda functions.

Code entry type Edit code inline Upload a .ZIP file Upload a .ZIP from Amazon S3

```
1 console.log('Loading function');
2
3 exports.handler = function(event, context) {
4     //console.log('Received event:', JSON.stringify(event, null, 2));
5     var message = event.Records[0].Sns.Message;
6     console.log('From SNS:', message);
7     context.succeed(message);
8 }
```

- 5.2.6 Scroll down to the **Lambda function handler and role** section, and from the drop-down list, choose the **Bootcamp2015-Lambda-Execution-Role** that we created previously.

Lambda function handler and role

Handler* index.handler

Role* Bootcamp2015-Lambda-Execution-Role

Ensure that popups are enabled to create a new role. [Learn more](#) about Lambda execution roles.

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)* 128

Timeout (s)* 3

* These fields are required.

Cancel Previous Next

Click the **Next** button.

- 5.2.7 Review the Lambda function and click the **Enable Now** option in the '**Enable event source**' section

Review

Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.

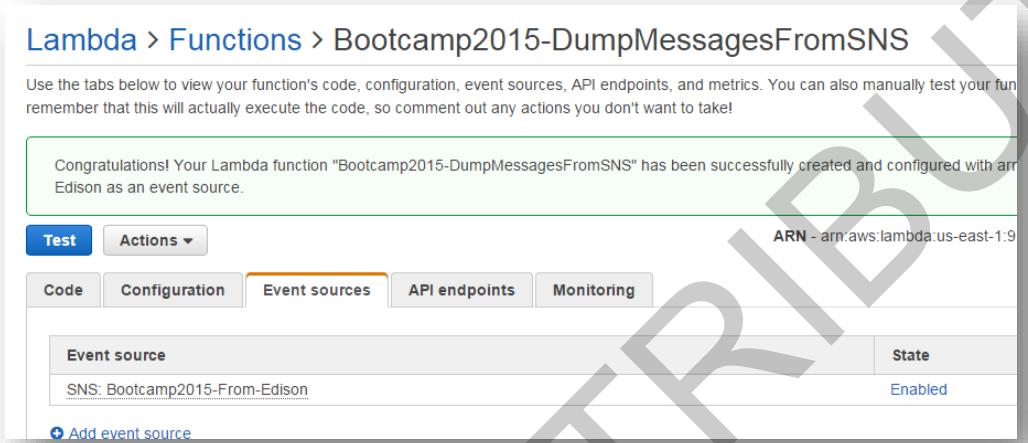
If the event source is created in an enabled state, it will begin invoking your function immediately. We recommend testing the function behavior in the next screen before enabling the event source. X

Event sources

Event source SNS: Bootcamp2015-From-Edison

Edit

Enable event source Enable now Enable later

5.2.8	<p>Scroll down to the 'Create Function' button and click it. You will see a success message:</p>  <p>The screenshot shows the Lambda Functions console with the path 'Lambda > Functions > Bootcamp2015-DumpMessagesFromSNS'. The 'Event sources' tab is selected. A green success message box says: 'Congratulations! Your Lambda function "Bootcamp2015-DumpMessagesFromSNS" has been successfully created and configured with an SNS topic named "SNS: Bootcamp2015-From-Edison" as an event source.' Below the tabs, there's a table with one row. The columns are 'Event source' and 'State'. The row contains 'SNS: Bootcamp2015-From-Edison' and 'Enabled'. There's also a blue '+ Add event source' button.</p>
5.2.9	<p>Now, again working in pairs, nominate one user to be the Primary and the other to be the Secondary</p> <p>You will again need to edit the primary's NodeJS test file so that it knows the URL of the SQS queue to send to in order to reach the secondary Edison.</p> <p>This is similar to what we did in an earlier task in this lab guide, however, the file we are going to edit is different to the previous file.</p>

5.2.10	<p>The file you need to edit on the Primary is ‘primary-send-sqs-to-initiate-pair-with-secondary.js’. To edit with vi run:</p> <pre>vi primary-send-sqs-to-initiate-pair-with-secondary.js</pre> <p>Again with the vi tool, press the “i” key, find the SQS_URL_QUEUE_TO_SECONDARY_EDISON variable and update this URL like you did before with the account ID of your partner.</p>  <p>Save the file in vi by pressing Esc, and then enter</p> <pre>:x</pre>
5.2.11	<p>On the primary, run</p> <pre>cd /home/root/labs node primary-send-sqs-to-initiate-pair-with-secondary.js</pre>
5.2.12	<p>On the secondary run:</p> <pre>cd /home/root/labs node secondary-announce-to-primary-on-sqs-received.js</pre>

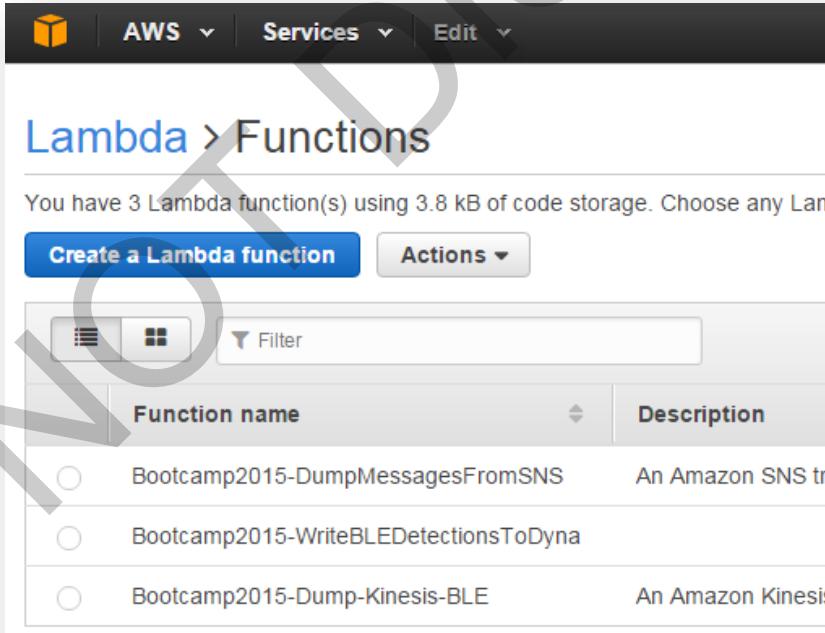
5.2.13	<p>Secondary</p> <pre>root@mini2:~# node 11-secondary-announce-to-primary-on-sqs-received.js onNobleStateChange -> poweredOn Starting to Advertise this Edison... Initialising Cognito... Cognito IdentityId => us-east-1:2710ba4d-1aa0-4683-aa6b-44fee6788f1d Starting to listen for other Edisons...</pre> <p>Primary</p> <pre>root@mini:~# node 11-primary-send-sqs-to-initiate-pair-with-secondary.js Initialising Cognito... Cognito IdentityId => us-east-1:65b6aaa0-6964-4b56-b3dd-ad372d080672 SQS message sent ok</pre>
5.2.14	<p>When the secondary receives the SQS message, you will see the following screens:</p> <p>Secondary</p> <pre>*** COMMAND_PAIR_REQUEST received from SQS -> Pair with 192.168.200.1 Attempting MQTT connection to 192.168.200.18 Connected Ok</pre> <p>Primary</p> <pre>Dispatching command (4) COMMAND_ANNOUNCE from 192.168.200.7 == > 192.168.200.7:1 SNS sending asynchronously SNS sent ok</pre>
5.2.15	<p>We will now browse the CloudWatch logs for the Lambda function you just executed. Under the Monitoring tab click on View Logs in CloudWatch</p> 

5.2.16	<p>Click the Log Stream and sort by Last Event Time and you should see the logs from the Lambda function that was just executed.</p> <pre data-bbox="355 439 1400 925"> Event Data ▼ 2015-07-05T09:44:53.747Z dae0cvhneu0ef1zi Loading function ▼ START RequestId: 7cbf2d05-22fa-11e5-bba9-c1c89bcd94b9 ▼ 2015-07-05T09:44:53.863Z 7cbf2d05-22fa-11e5-bba9-c1c89bcd94b9 Received event: { "Records": [{ "EventSource": "aws:sns", "EventVersion": "1.0", "EventSubscriptionArn": "arn:aws:sns:us-east-1:532236681459:Bootcamp2015-From-Edison:b19bf8b1-cd7f-4f92-810e-f7" "Sns": { "Type": "Notification", "MessageId": "0b4cf755-6e5e-5878-9f2b-b947908b7677", "TopicArn": "arn:aws:sns:us-east-1:532236681459:Bootcamp2015-From-Edison", "Subject": "COMMAND_PAIR_REQUEST from your Edison", "Message": "{\"eventDate\":\"2015-07-05T09:44:51.803Z\", \"command\":2, \"senderIpAddress\":\"192.168.200.18\\Address\":\"192.168.200.71\", \"announcedDeviceType\":\"1\"}", "Timestamp": "2015-07-05T09:44:53.427Z", "SignatureVersion": "1", "Signature": "gsv0rHwgzCE+5D21+RV+5o+XtMYBzlw8YHNBUyVmNQAr0/bx8n/5G/+DADbS/bb1balwA/p3cZFyRikVlIt+Raax1g6J.iwEm1DrLh5KbefMEnD23Fk8CBGIVwn1t0kTBQprytpgv4mxm1k69MLqysj837DTxZ0scvYS36qo381fVYcpT1Id1Hm0B9yAFH+NX6cSCBXgezRM8nQ6YrZpEW.ro9WLieL3PALWEk5oI5drueRapRH2rvxf1a0j5BMDsKECLZEK+TEa+P9zDIBGmXG+EjHeUh1huUYaC8iZ4Q==", "SigningCertUrl": "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-d6d679a1d18e95c2f9fffc11f4-otcamp2015-From-Edison:b19bf8b1-cd7f-4f92-810e-f77c7ae7c05e", "UnsubscribeUrl": "https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-e" "MessageAttributes": {} } }] } </pre>
5.2.17	<p>After you have viewed the logs and understand how this part works, you are ready to move to the next task, where we will next create the final Lambda function that will do the work in our system.</p>

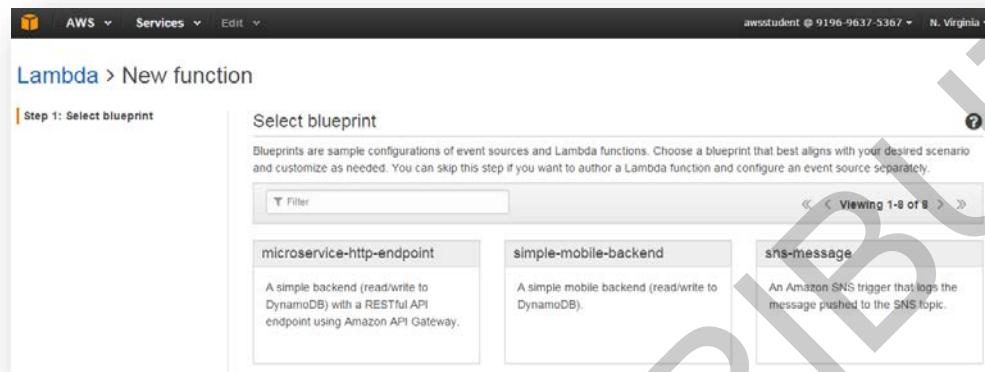
Task 6: Dispatching commands from Edison via SNS to Companion App

Overview	<p>In this module, we will dispatch the messages sent from Edison to Lambda, via SNS, to perform various tasks. This follows on from the previous module, where we simply dumped out the SNS notifications to the log. Here, we will create a new Lambda function and the code we run in this function will perform all the tasks of our final application.</p>
-----------------	---

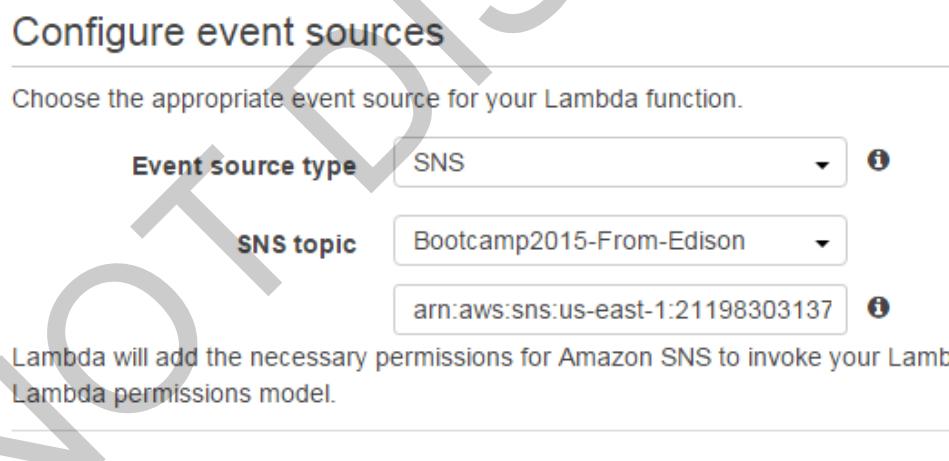
Task 6-1: Create a new Lambda Function

Step	Instruction								
6.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Compute- Lambda.</p>  <p>The screenshot shows the AWS Management Console navigation bar with 'AWS' and 'Services' dropdowns, and an 'Edit' button. Below the navigation bar, there's a grid of service icons: History (Lambda), All AWS Services (Compute, SNS, SQS), EC2 (Amazon Elastic Compute Cloud provides resizable compute capacity in the cloud), Lambda (AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.), and EC2 Container Service (Amazon ECS allows you to easily run and manage Docker containers across a cluster of Amazon EC2 Instances).</p>								
6.1.2	<p>The Lambda console is displayed here.</p>  <p>The screenshot shows the Lambda > Functions page. The top navigation bar includes 'AWS', 'Services', 'Edit', and a user profile. Below the navigation is a header with 'Lambda > Functions' and a message stating 'You have 3 Lambda function(s) using 3.8 kB of code storage. Choose any Lambda function to edit it.' There are 'Create a Lambda function' and 'Actions' buttons. A table lists three Lambda functions: 'Bootcamp2015-DumpMessagesFromSNS', 'Bootcamp2015-WriteBLEDetectionsToDyna', and 'Bootcamp2015-Dump-Kinesis-BLE'. Each row in the table has a small icon, a 'Function name' column, and a 'Description' column.</p> <table border="1"><thead><tr><th>Function name</th><th>Description</th></tr></thead><tbody><tr><td>Bootcamp2015-DumpMessagesFromSNS</td><td>An Amazon SNS trigger</td></tr><tr><td>Bootcamp2015-WriteBLEDetectionsToDyna</td><td></td></tr><tr><td>Bootcamp2015-Dump-Kinesis-BLE</td><td>An Amazon Kinesis trigger</td></tr></tbody></table>	Function name	Description	Bootcamp2015-DumpMessagesFromSNS	An Amazon SNS trigger	Bootcamp2015-WriteBLEDetectionsToDyna		Bootcamp2015-Dump-Kinesis-BLE	An Amazon Kinesis trigger
Function name	Description								
Bootcamp2015-DumpMessagesFromSNS	An Amazon SNS trigger								
Bootcamp2015-WriteBLEDetectionsToDyna									
Bootcamp2015-Dump-Kinesis-BLE	An Amazon Kinesis trigger								

6.1.3 Click **Create a Lambda Function**, then choose the **sns-message** blueprint:



6.1.4 In the Configure Event Sources section, select the SNS topic **Bootcamp2015-From-Edison** that we just created in the previous task.



Click the **Next** button

6.1.5	<p>In the Configure Function section, name the new Lambda function Bootcamp2015-DispatchCommandsFromSNS</p> <p>Configure function</p> <p>A Lambda function consists of the custom code you want to execute. Learn more about Lambda functions.</p> <table border="1"> <tr> <td>Name*</td> <td>Bootcamp2015-DispatchCommandsFr</td> </tr> <tr> <td>Description</td> <td>An Amazon SNS trigger that logs the message pushed to the topic</td> </tr> <tr> <td>Runtime*</td> <td>Node.js</td> </tr> </table>	Name*	Bootcamp2015-DispatchCommandsFr	Description	An Amazon SNS trigger that logs the message pushed to the topic	Runtime*	Node.js
Name*	Bootcamp2015-DispatchCommandsFr						
Description	An Amazon SNS trigger that logs the message pushed to the topic						
Runtime*	Node.js						
6.1.6	<p>In the Lambda function code section, remove the template code that is present, and instead paste in the contents of the Bootcamp2015-DispatchCommandsFromSNS.lambda.js file which is in the bundle you downloaded in the previous lab guides and referred to as the Desktop bundle in previous labs.</p>						

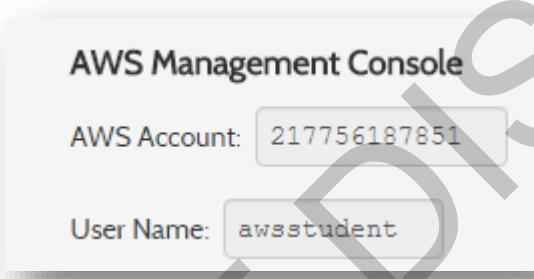
6.1.7

In the code you just pasted, scroll down to line 55

```
49 //////////////////////////////////////////////////////////////////
50 //
51 // CONFIGURATION: Add the Url to the SNS topic 'Bootcamp2015-To-MobileApp'
52 //
53 //////////////////////////////////////////////////////////////////
54
55 var SNS_TOPIC_TO_MOBILE_APP = "arn:aws:sns:us-east-1:XXXXXXXXXX:Bootcamp2015-To-MobileApp";
56 var TABLE_NAME_PAIR_LIST = "Bootcamp2015-BLEDetections";
57
```

In the value set for SNS_TOPIC_TO_MOBILE_APP, you will notice a series of 'XXXX'. This is a placeholder for you to add in your AWS Account Id

You can get your account number from the Qwiklabs dashboard, where you started the Qwiklab from. An example of the Account Id is shown below:

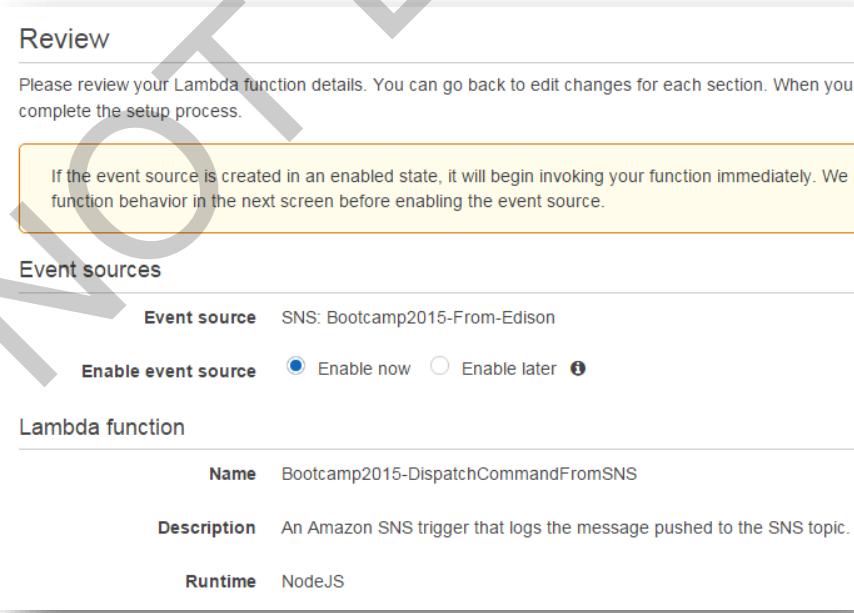


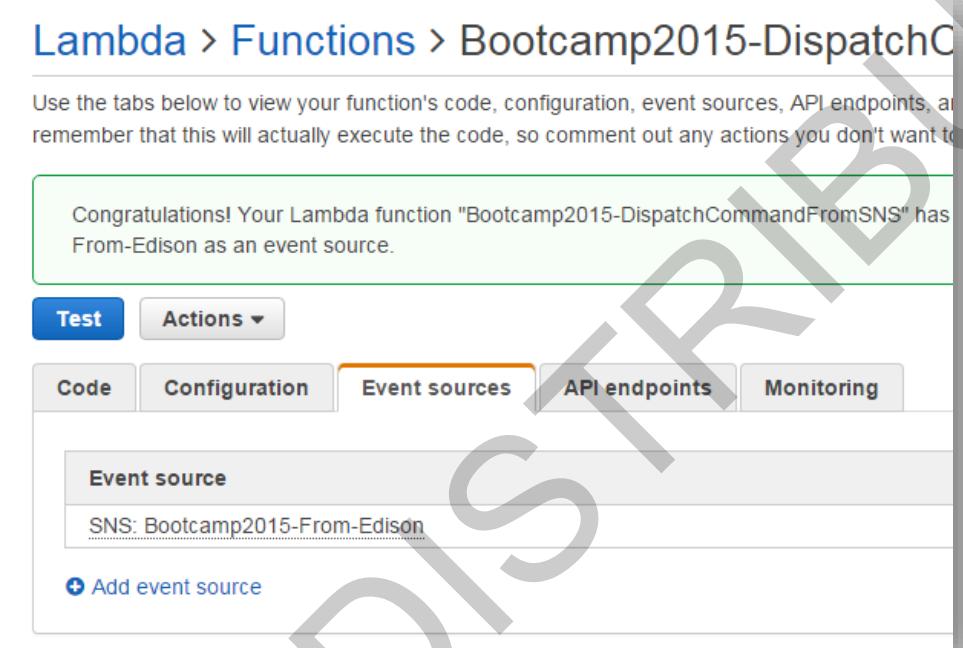
Edit the line in the Lambda function so that it contains your Account Id. The result will look something like this:

```
var SNS_TOPIC_TO_MOBILE_APP = "arn:aws:sns:us-east-1:448267577043:Bootcamp2015-To-MobileApp";
```

Note: Your account Id will be different! You cannot use the account Id shown above, you must use the account Id for the Qwiklabs account you are using for the Bootcamp!

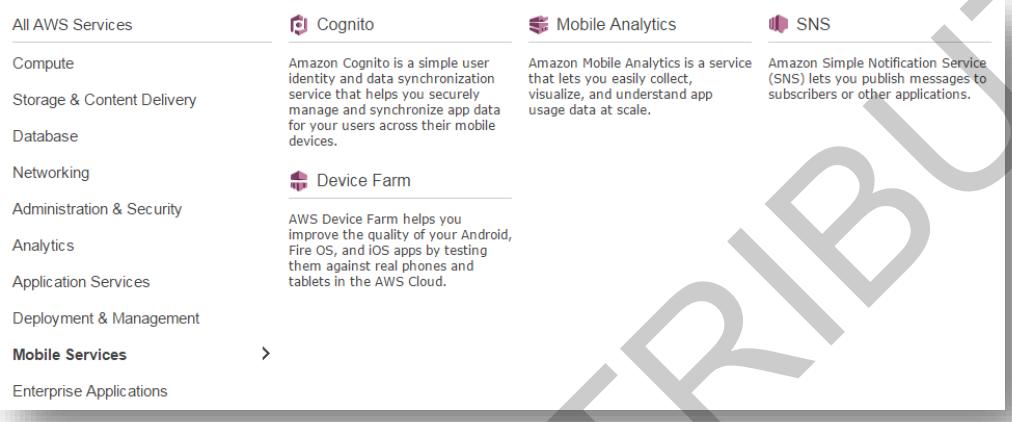
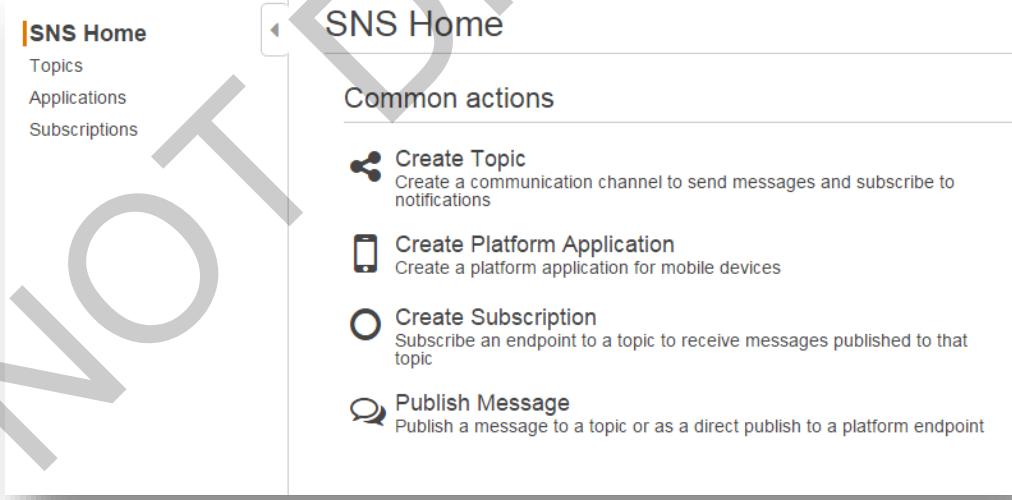
We have not yet created the topic Bootcamp2015-To-MobileApp but we will do this later in this Lab guide

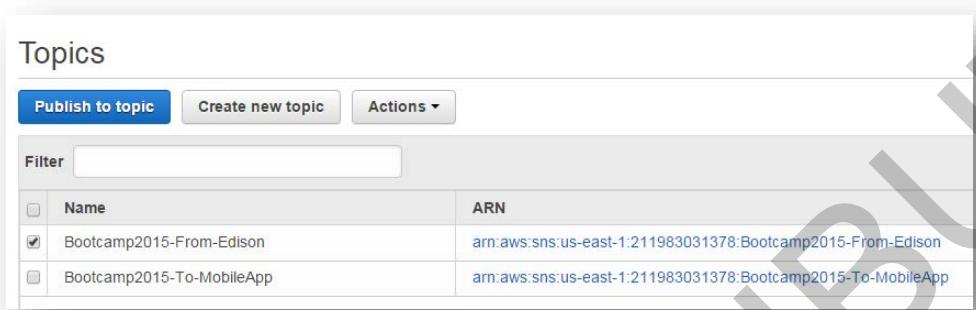
6.1.8	<p>Scroll down to the Lambda function handler and role section.</p> <p>Again, choose the Bootcamp2015-Lambda-Execution-Role for the role</p> <p>In the Advanced Setting section, increase the Timeout to 15 seconds. We will need additional time when making other calls from this Lambda function later</p> 
6.1.9	<p>Click Next to review your function.</p> <p>Ensure that you select the Enable now option in the Enable event source section:</p> 

6.1.10	<p>Scroll down to find the Create Function button, and click it.</p> <p>Your function will be created:</p>  <p>The screenshot shows the Lambda Functions console with the path 'Lambda > Functions > Bootcamp2015-DispatchCommandFromSNS'. Below the path, a message says 'Congratulations! Your Lambda function "Bootcamp2015-DispatchCommandFromSNS" has From-Edison as an event source.' A navigation bar at the top includes 'Test' and 'Actions' buttons, followed by tabs for 'Code', 'Configuration', 'Event sources' (which is highlighted in orange), 'API endpoints', and 'Monitoring'. Under the 'Event sources' tab, there is a section titled 'Event source' containing the entry 'SNS: Bootcamp2015-From-Edison'. At the bottom of this section is a blue '+ Add event source' button.</p>
6.1.11	Now, whenever an SNS notification is sent to the Bootcamp2015-From-Edison topic, your Lambda function will be called to process the message.
6.1.12	Later in the Bootcamp, we will be sending messages via this SNS Topic from the Edison. For the moment, we will manually test if everything is working by using the AWS Console.

Task 6-2: Test the new Lambda Function and SNS Event Source

Step	Instruction

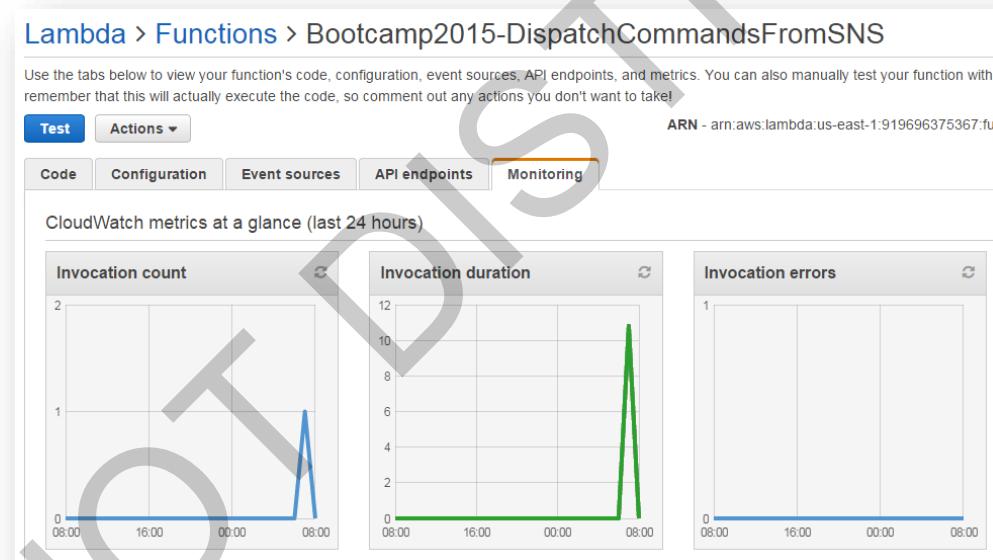
6.2.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Mobile Services – SNS.</p>  <p>All AWS Services</p> <ul style="list-style-type: none"> Compute Storage & Content Delivery Database Networking Administration & Security Analytics Application Services Deployment & Management Mobile Services Enterprise Applications <p>Cognito Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices.</p> <p>Mobile Analytics Amazon Mobile Analytics is a service that lets you easily collect, visualize, and understand app usage data at scale.</p> <p>SNS Amazon Simple Notification Service (SNS) lets you publish messages to subscribers or other applications.</p>
6.2.2	<p>On the SNS dashboard in the AWS console, click Topics on the left navigation bar:</p>  <p>SNS Home</p> <p>Topics</p> <p>Common actions</p> <ul style="list-style-type: none"> Create Topic Create a communication channel to send messages and subscribe to notifications Create Platform Application Create a platform application for mobile devices Create Subscription Subscribe an endpoint to a topic to receive messages published to that topic Publish Message Publish a message to a topic or as a direct publish to a platform endpoint

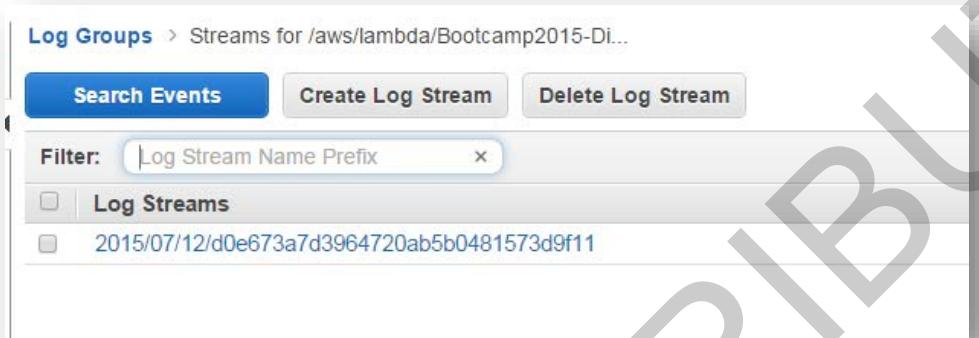
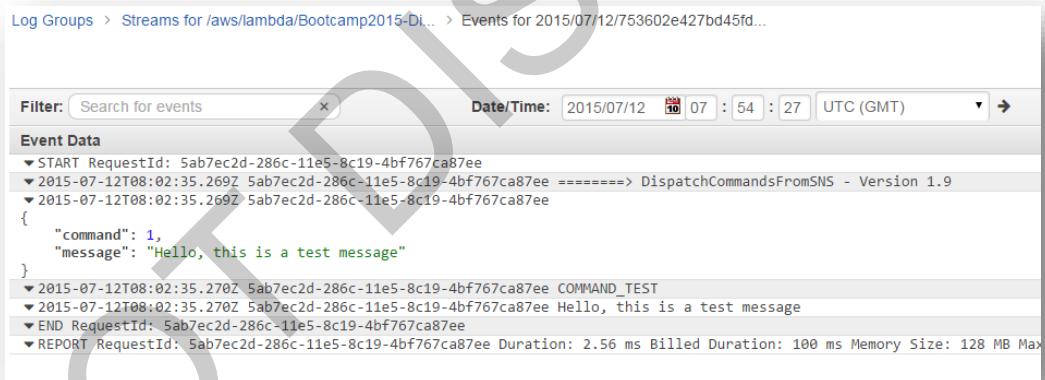
6.2.3	<p>Select the Bootcamp2015-From-Edison topic, and click Publish to Topic.</p> 
6.2.4	<p>In the Message section, paste in the following text:</p> <pre>{ "command": 1, "message": "Hello, this is a test message" }</pre> 
6.2.5	<p>Leave the other fields as defaults, and click the Publish Message button. This will send a test message to the SNS topic</p>

- 6.2.6 Login to the AWS Management Console from your Qwiklabs Environment and choose **Services – Compute – Lambda**.



- 6.2.7 In the Lambda console for **Bootcamp2015-DispatchCommandsFromSNS**, click the **Monitoring** tab:



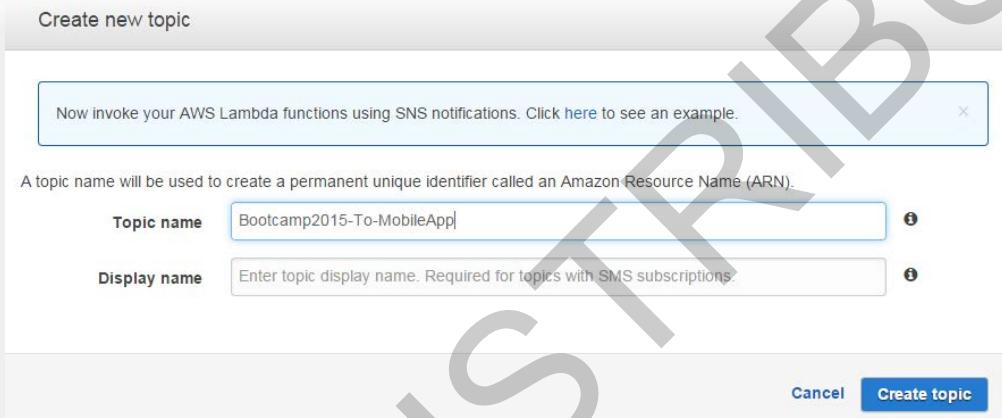
6.2.8	<p>Notice that there has been activity on the Lambda function. Click the View Logs in Cloudwatch link. The CloudWatch log stream will open.</p> 
6.2.9	<p>Select the log stream, and view the details.</p>  <p>Notice how the message you sent has been received and processed.</p>

Task 6-3: Create new SNS Topic for the Mobile app

Overview	<p>We will create a new SNS topic that the companion mobile app will subscribe to at runtime.</p> <p>The Lambda function we created previously (<i>Bootcamp2015-DispatchCommandsFromSNS</i>) will push messages to this topic in order to notify the companion mobile app of an event, like a pairing or a message from another user.</p> <p>We do not need a Lambda function bound to this Topic.</p>
----------	---

Step	Instruction
6.3.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Mobile Services -> SNS button.</p>  A screenshot of the AWS Management Console navigation bar. The 'Mobile Services' option is highlighted in blue, indicating it is selected. Other options visible include All AWS Services, Compute, Storage & Content Delivery, Database, Networking, Administration & Security, Analytics, Application Services, Deployment & Management, and Enterprise Applications. To the right of the navigation bar, there are four service cards: Cognito, Mobile Analytics, Device Farm, and SNS. The SNS card is the one that has been selected.

- 6.3.2 In the SNS console, click **Create Topic**, name it
Bootcamp2015-To-MobileApp
and click **Create Topic**.



Lab 7: Implementing Developer Authentication for our Mobile app, using API Gateway and Lambda functions

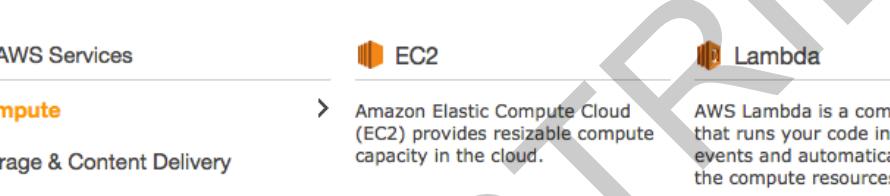
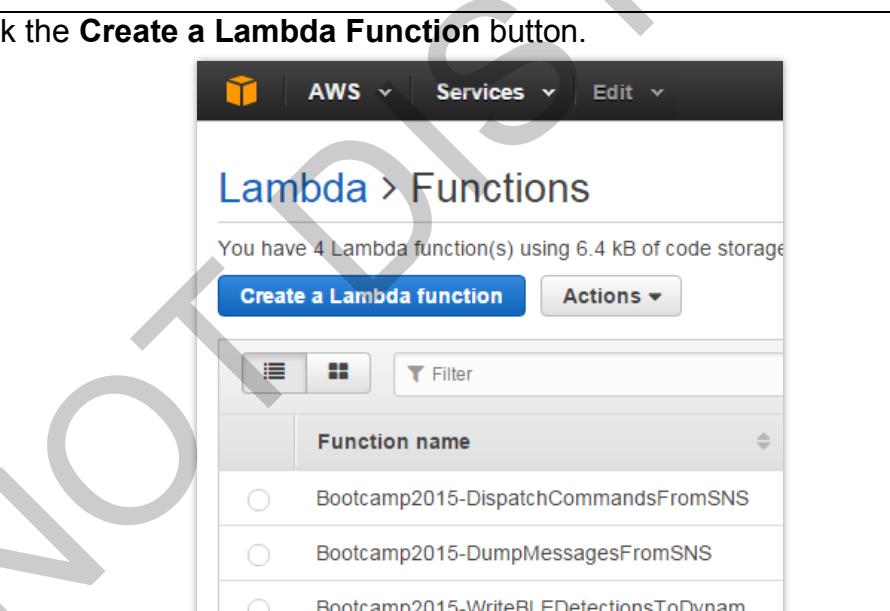
Overview	In this lab you will learn how to create the two methods we need to implement our own Developer Authentication through Cognito, using Amazon Lambda functions via the Amazon API Gateway.
Objectives	<p>After completing this lab, you will be able to:</p> <ul style="list-style-type: none">• Create Lambda functions that are exposed via a RESTful endpoint using the Amazon API Gateway, including relevant request and response data models• Implement a simple login function in Lambda• Implement a simple lambda function that exposes the Cognito function <code>getOpenIdTokenForDeveloperIdentity()</code> from the AWS SDK for NodeJS• Run simple tests from the AWS Console to confirm that the API Gateway and Lambda functions are working as expected.
Pre-requisites	<p>This lab requires:</p> <ul style="list-style-type: none">• Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">◦ Note: The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.• For Microsoft Windows users: Administrator access to the computer.• An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).• An SSH client, such as PuTTY.• At least 2 available USB ports on your notebook computer.
Duration	30 minutes

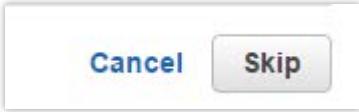
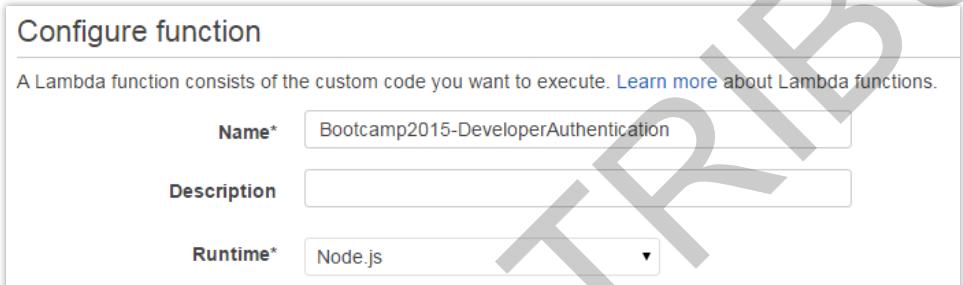
Task 1: Create two new Lambda Functions for authentication via Amazon Cognito

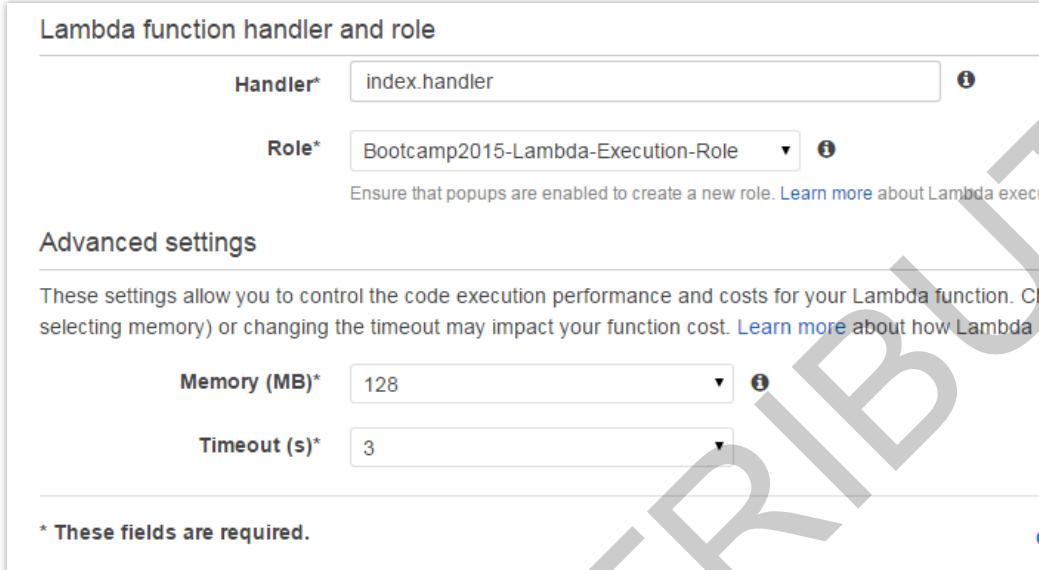
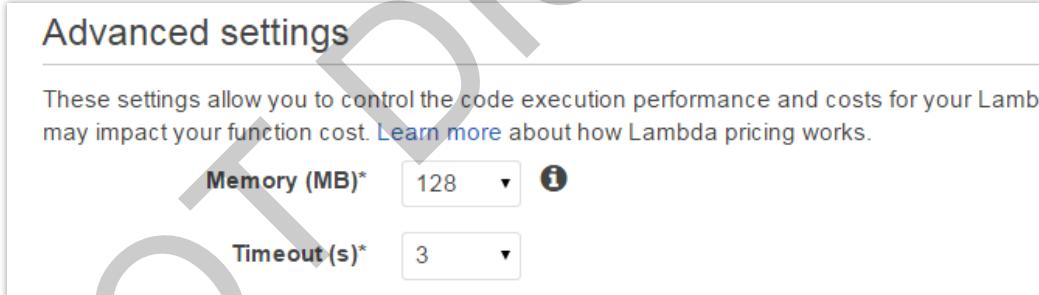
Overview	<p>To implement Developer Authentication, we need to expose two methods to our Bootcamp application:</p> <ol style="list-style-type: none">Login: This function will be called to authenticate the username/password combination provided by the userGetToken: This function will be called by our implementation in the mobile app, in order to get or refresh its copy of the authentication token required by Cognito <p>We will implement these two functions using Amazon Lambda, and expose them to our mobile application via the Amazon API Gateway. First, we need to create the two Lambda functions.</p>
-----------------	--

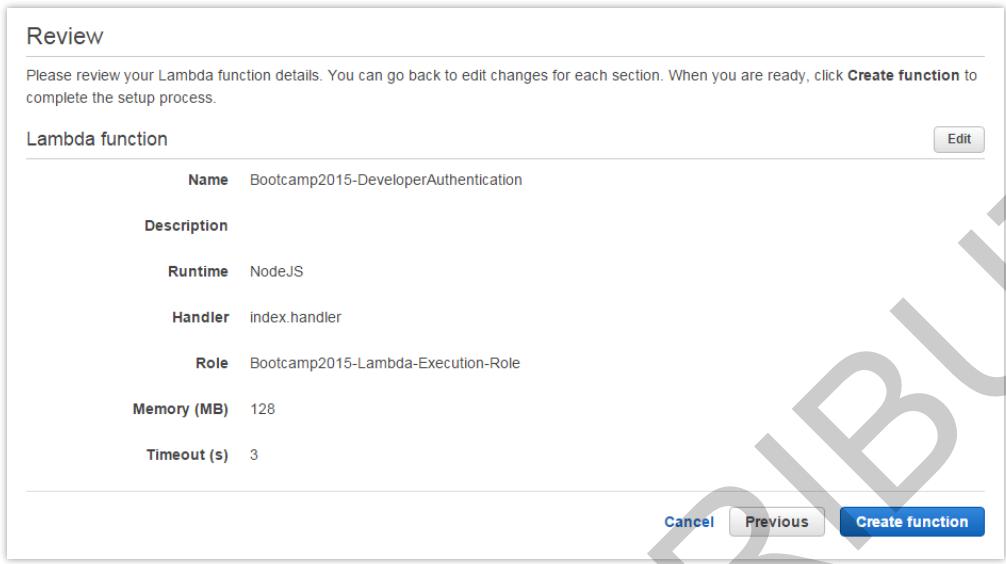
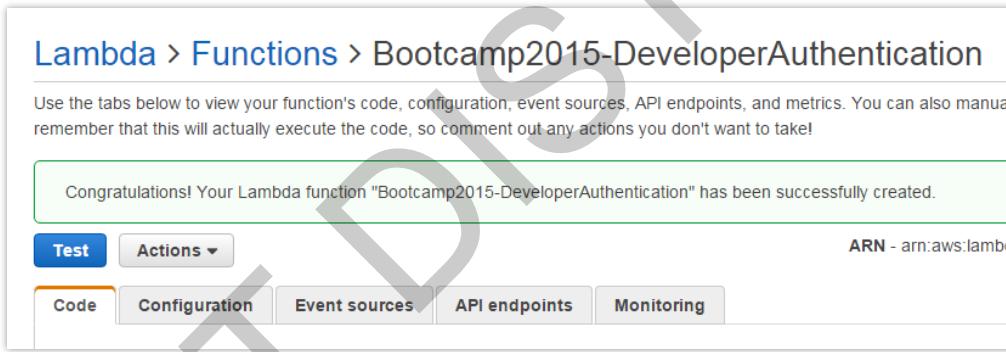
Task 1-1: Create the *Bootcamp2015-DeveloperAuthentication* Lambda Function

Lambda Function	
Overview	In this section we will set up and create the Bootcamp2015-DeveloperAuthentication Lambda function that we will be using to authenticate our users.

Step	Instruction
1.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services - Compute – Lambda.</p>  <p>The screenshot shows the AWS navigation bar with 'All AWS Services' selected. Below it, 'Compute' is highlighted in orange, indicating the current service. To the right of 'Compute' is a blue arrow pointing to the 'Lambda' service, which is also highlighted in orange. The Lambda service page is displayed, featuring its logo, name, and a brief description: 'AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.'</p>
1.1.2	<p>Click the Create a Lambda Function button.</p>  <p>The screenshot shows the 'Lambda > Functions' page. At the top, there is a message: 'You have 4 Lambda function(s) using 6.4 kB of code storage'. Below this is a blue 'Create a Lambda function' button. To the right of the button is a 'Actions' dropdown menu. The main area displays a table with a single column labeled 'Function name'. Four functions are listed: 'Bootcamp2015-DispatchCommandsFromSNS', 'Bootcamp2015-DumpMessagesFromSNS', 'Bootcamp2015-WriteBLEDetectionsToDynam...', and 'Bootcamp2015-Dump-Kinesis-BLE'. Each function name is preceded by a small circular checkbox.</p>

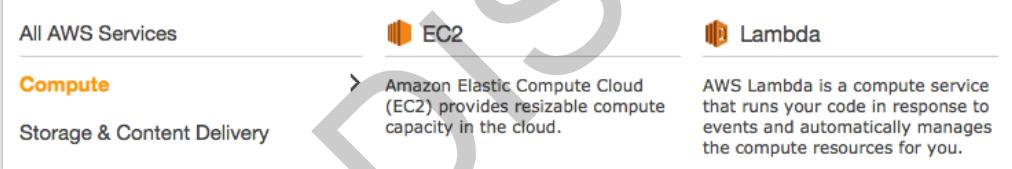
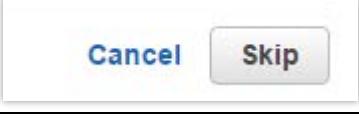
1.1.3	<p>You will see the Blueprint panel.</p> <p>We will not use a blueprint for our function, so scroll down to the bottom, and on the right, locate and click the Skip button.</p> 
1.1.4	<p>On the Configure Function page, name the Lambda Function</p> <p>Bootcamp2015-DeveloperAuthentication</p> 
1.1.5	<p>Scroll down to the Lambda function code section. In the text edit window, paste the contents of the file</p> <p><i>Bootcamp2015-DeveloperAuthentication.lambda.js</i></p> <p>which you will find in your Bootcamp Desktop bundle, that you downloaded in Lab 1.</p>  <pre> 41 // Log some stuff 42 // 43 console.log(g_logTag + 'event: ' + JSON.stringify(event)); 44 // 45 // Authenticate this username/password - you would call out to LDAP or your own 46 // password database for this - this is only for demonstration! 47 // 48 if (event.username === "username" && event.password === "password") 49 { 50 context.succeed({ succeeded: true }); 51 console.log("Authentication succeeded"); 52 } 53 else 54 { 55 // 56 // FAILED to authenticate! 57 // 58 context.fail(new Error('BootcampIdP has failed to authorise you due to username/password mismatch')); 59 } 60 } 61 </pre>

1.1.6	<p>Scroll down below the code section, to the Lambda function handler and role section.</p>  <p>Lambda function handler and role</p> <p>Handler*: index.handler</p> <p>Role*: Bootcamp2015-Lambda-Execution-Role</p> <p>Ensure that popups are enabled to create a new role. Learn more about Lambda execution roles.</p> <p>Advanced settings</p> <p>These settings allow you to control the code execution performance and costs for your Lambda function. Changing memory or selecting memory (or changing the timeout) may impact your function cost. Learn more about how Lambda pricing works.</p> <p>Memory (MB)*: 128</p> <p>Timeout (s)*: 3</p> <p>* These fields are required.</p>
1.1.7	<p>Choose the Bootcamp2015-Lambda-Execution-Role; this is a role we created in an earlier lab guide.</p>
1.1.8	<p>Leave the Advanced settings as they are; they should be adequate for this module.</p>  <p>Advanced settings</p> <p>These settings allow you to control the code execution performance and costs for your Lambda function. Changing memory or selecting memory (or changing the timeout) may impact your function cost. Learn more about how Lambda pricing works.</p> <p>Memory (MB)*: 128</p> <p>Timeout (s)*: 3</p>

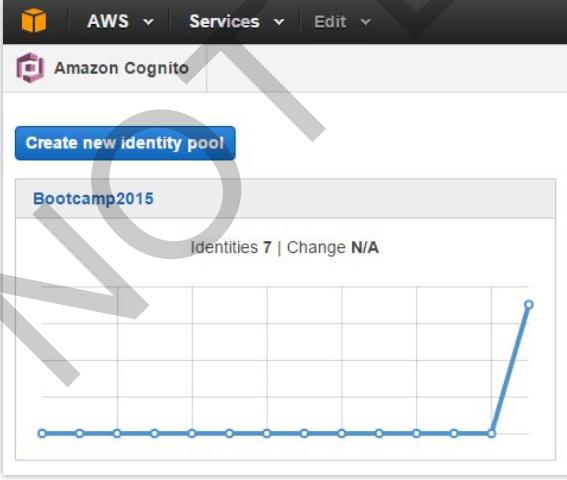
1.1.9	<p>Click the Next button to review the function setup.</p> 
1.1.10	<p>Click the Create function button to create the new function.</p>
1.1.11	<p>The Lambda function will be created, and you will see a success message.</p> 

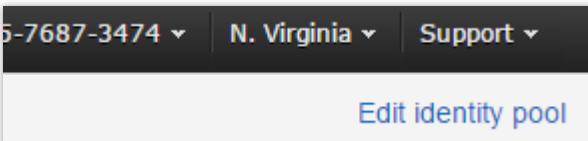
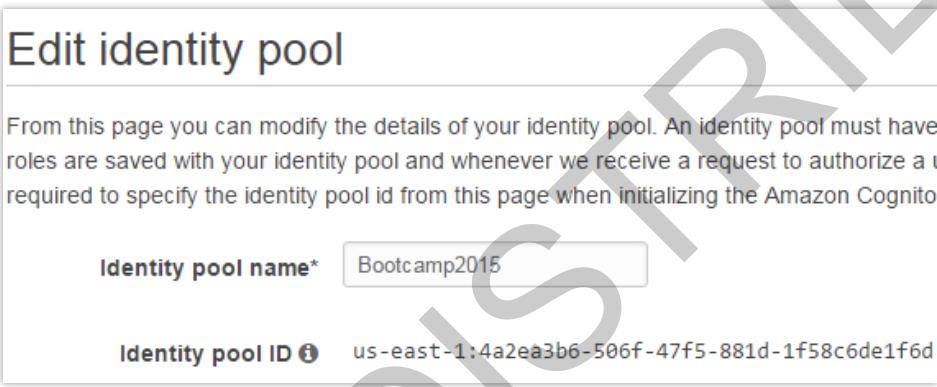
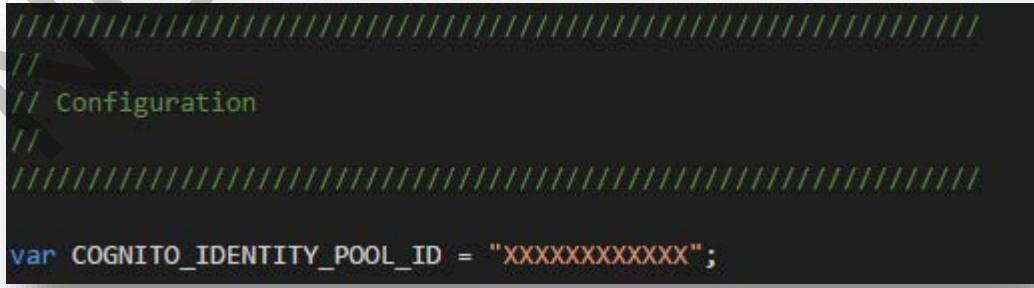
Task 1-2: Create the Bootcamp2015-DeveloperAuthGetToken Lambda Function

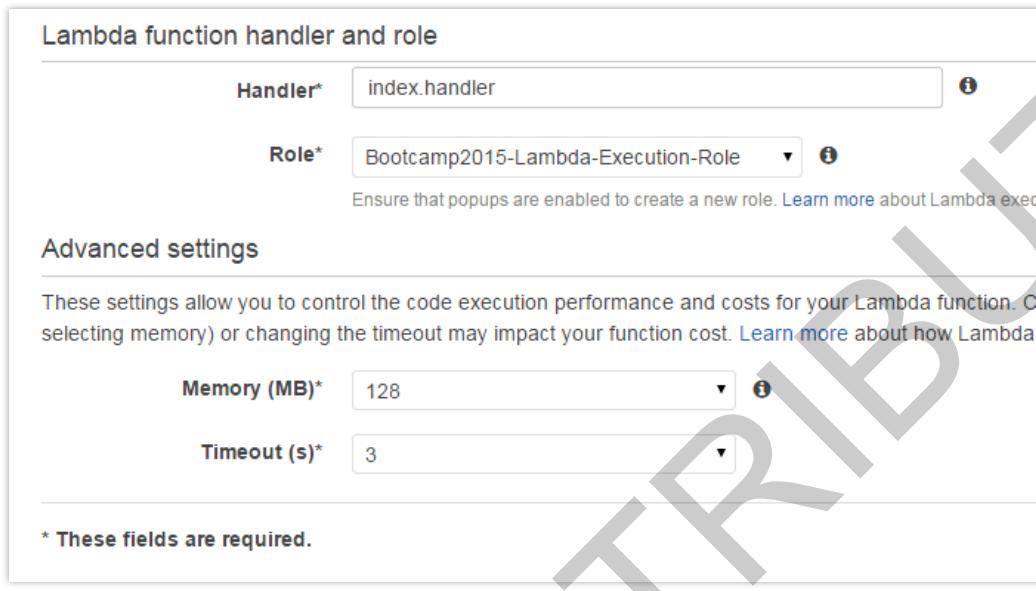
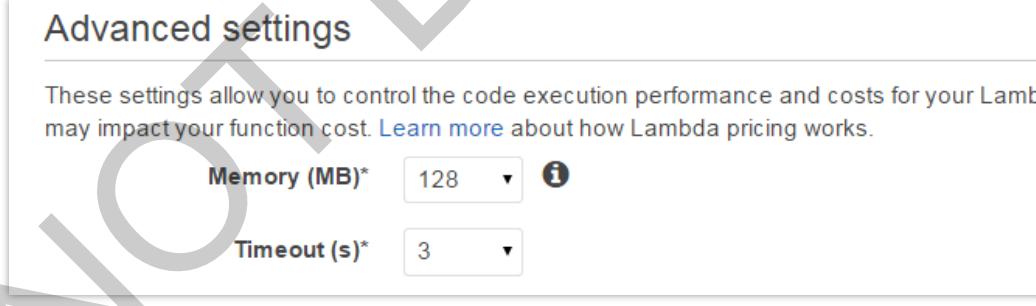
Overview	In this section we will setup and create the Bootcamp2015-DeveloperAuthGetToken Lambda function which we will be using to refresh the token that we use to authenticate our users.
-----------------	--

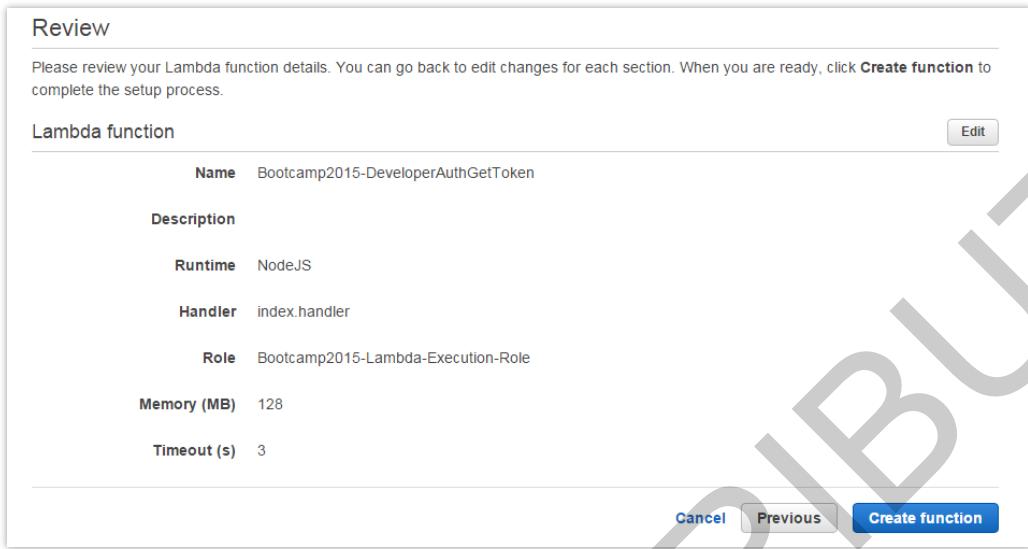
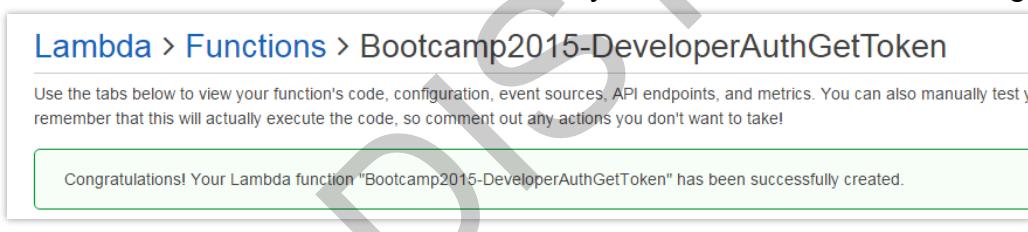
Step	Instruction
1.2.1	<p>If you are following on from the previous task, you can simply click the ‘Functions’ link at the top of the page to get back to the Lambda dashboard:</p> <p style="text-align: center;">Lambda > Functions > Bootcamp2015-DeveloperAuthentication</p> <p>If you are commencing this task from any other location, launch the AWS Management Console from your Qwiklabs environment and choose Services – Compute – Lambda.</p>  <p>AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.</p>
1.2.2	<p>Click the Create a Lambda Function button.</p> 
1.2.3	<p>You will see the Blueprint panel.</p> <p>We will not use a blueprint for our function, so scroll down to the bottom, and on the right, locate and click the Skip button:</p> 

1.2.4	<p>On the Configure Function page, name the Lambda Function Bootcamp2015-DeveloperAuthGetToken</p> <p>Configure function</p> <p>A Lambda function consists of the custom code you want to execute. Learn more about Lambda functions.</p> <p>Name* <input type="text" value="Bootcamp2015-DeveloperAuthGetToken"/></p> <p>Description <input type="text"/></p> <p>Runtime* <input type="text" value="Node.js"/></p>
1.2.5	<p>Scroll down to the Lambda function code section. In the text edit window, paste in the contents of the file Bootcamp2015-DeveloperAuthGetToken.lambda.js which you will find in your Bootcamp Desktop bundle.</p> <p>Lambda function code</p> <p>Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk). If you need custom libraries, you can upload your code and libraries as a .ZIP file. Learn more about deploying Lambda functions.</p> <p>Code entry type <input checked="" type="radio"/> Edit code inline <input type="radio"/> Upload a .ZIP file <input type="radio"/> Upload a .ZIP from Amazon S3</p> <pre>112 console.log(err, err.stack); // an error occurred 113 console.log(JSON.stringify(result)); 114 context.fail(result); 115 } 116 else { 117 var result = { 118 "errorMessage": "", 119 "errorType": 0, 120 "identityId": data.IdentityId, 121 "token": data.Token 122 }; 123 124 console.log(JSON.stringify(result)); 125 context.succeed(result); 126 127 }); 128 } 129 130 }; 131 }; 132 }</pre>

1.2.6	<p>You will need to edit this code and paste in the Cognito Identity Pool Id in line 32, in the configuration section:</p> <pre>// Configuration // ... var COGNITO_IDENTITY_POOL_ID = "XXXXXXXXXXXXXX";</pre> <p>You can retrieve your Cognito Identity Pool Id from the bootcamp-config.js file, because you previously edited that file to add it in.</p> <p>Or, if you prefer, you can browse to the Cognito Console and retrieve the Id from there. The next few steps will guide you through that process. You can skip these if you already have your Cognito Identity Pool Id handy.</p>
1.2.7	<p>How to retrieve Your Cognito Identity Pool Id from the Console</p> <p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services -> Mobile Services -> Cognito.</p> <p>Click the Bootcamp2015 heading to open the Identity Pool.</p> 

1.2.8	<p>How to retrieve Your Cognito Identity Pool Id from the Console (continued)</p> <p>In the top right of the screen, click Edit Identity Pool</p> 
1.2.9	<p>How to retrieve Your Cognito Identity Pool Id from the Console (continued)</p> <p>Copy the full Identity Pool ID value shown at the top of the next page:</p>  <p>It should be in the form:</p> <p>us-east-1:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx</p>
1.2.10	<p>When you have your Cognito Identity Pool Id in the clipboard, paste this value into the Lambda function code to replace the “XXXXXX” section as shown below:</p>  <pre> // Configuration var COGNITO_IDENTITY_POOL_ID = "XXXXXXXXXXXX"; </pre>

1.2.11	<p>After you update the Lambda function code to add in the Cognito Identity Pool Id, scroll down below the code section, to the Lambda function handler and role section.</p>  <p>Lambda function handler and role</p> <p>Handler* index.handler</p> <p>Role* Bootcamp2015-Lambda-Execution-Role</p> <p>Ensure that popups are enabled to create a new role. Learn more about Lambda execution roles.</p> <p>Advanced settings</p> <p>These settings allow you to control the code execution performance and costs for your Lambda function. Changing memory or selecting a different timeout may impact your function cost. Learn more about how Lambda pricing works.</p> <p>Memory (MB)* 128</p> <p>Timeout (s)* 3</p> <p>* These fields are required.</p>
1.2.12	<p>Choose Bootcamp2015-Lambda-Execution-Role</p> <p>This is a role we created in an earlier lab guide.</p>
1.2.13	<p>Leave the Advanced settings as they are; they should be adequate for this module.</p>  <p>Advanced settings</p> <p>These settings allow you to control the code execution performance and costs for your Lambda function. Changing memory or selecting a different timeout may impact your function cost. Learn more about how Lambda pricing works.</p> <p>Memory (MB)* 128</p> <p>Timeout (s)* 3</p>

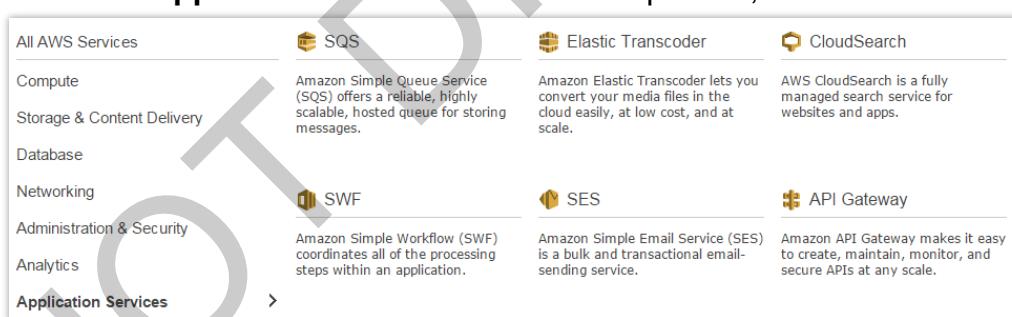
1.2.14	<p>Click the Next button to review the function setup.</p> 
1.2.15	<p>Click the Create function button to create the new function.</p>
1.2.16	<p>The Lambda function will be created, and you will see a success message.</p> 

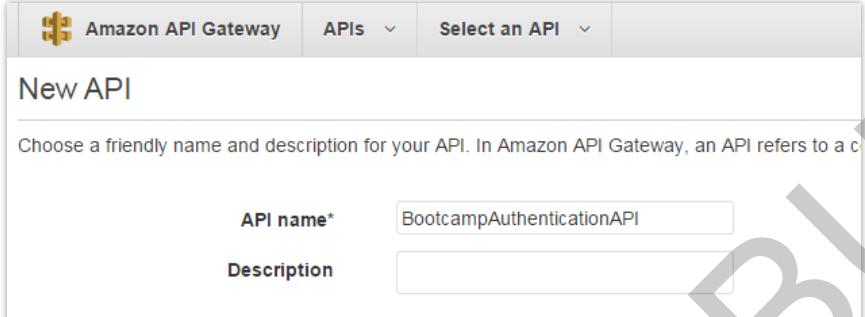
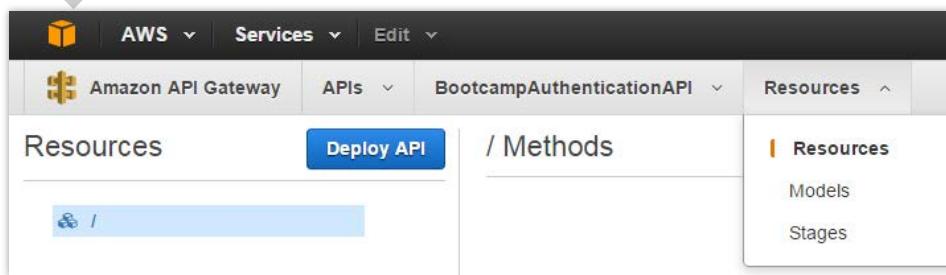
Task 2: Setting up the API Gateway to expose the two new Lambda functions to our mobile app

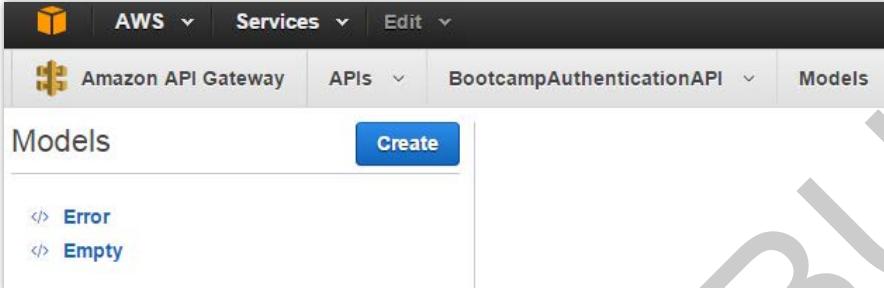
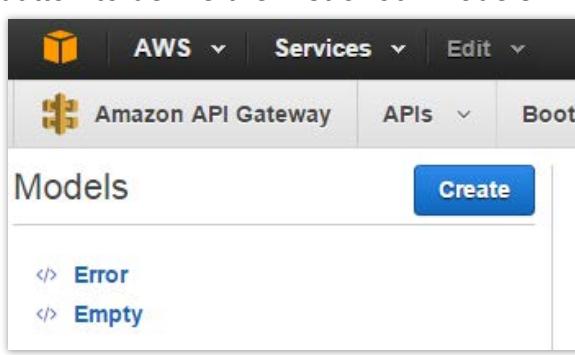
Overview	In this task, we will configure the Amazon API Gateway service to expose the two Lambda functions we just created, as RESTful endpoint. This will make them accessible to our Mobile application, in later Labs
-----------------	---

Task 2-1: Create the setup Amazon API Gateway

Overview	In this task, we will configure the API Gateway.
-----------------	--

Step	Instruction
2.1.1	<p>Launch the AWS Management Console from your Qwiklabs environment and under Application Services from the drop down, click API Gateway.</p> 
2.1.2	<p>Click Get started.</p> 

2.1.3	<p>Enter a name for your API. For this Bootcamp use</p> <p>BootcampAuthenticationAPI</p> 
2.1.4	<p>Click the Create API button</p> <p>Your new API will be created. You will see the following screen, indicating that there are no resources for this API and no methods defined, which is normal.</p> 
2.1.5	<p>We will create methods for our authentication API in a moment, but first, we will create Models that define the data exchange for our API</p> <p>Click the menu item that currently says Resources in order to access the navigation menu.</p> <p>From the navigation menu, select Models.</p> 

2.1.6	<p>There are two models, by default, included with every API:</p> <ul style="list-style-type: none"> • Error and • Empty  <p>We will leave these models as they are and create four new models that will define the classes that we use in our mobile application to send requests and interpret responses to/from our API. The API Gateway uses these models to define the data interfaces, and also, importantly, to create the SDK for our API Gateway. The four models are:</p> <ul style="list-style-type: none"> • <i>GetTokenRequestModel</i> is used for making a request against our API to get a token for our Developer Authenticated users • <i>GetTokenResponseModel</i> will define the response we get back from the <i>GetTokenRequestModel</i> call we made • <i>LoginRequestModel</i> is used for making a request against our API to authenticate a user • <i>LoginResponseModel</i> will define the response we get back from the <i>LoginResponseModel</i> call we made
2.1.7	<p>Click the Create button to define the first of our models:</p>  <p>We will now go through each of the four models outlined below.</p>

2.1.8

GetTokenRequestModel

As per the screen shot below, enter the following details.

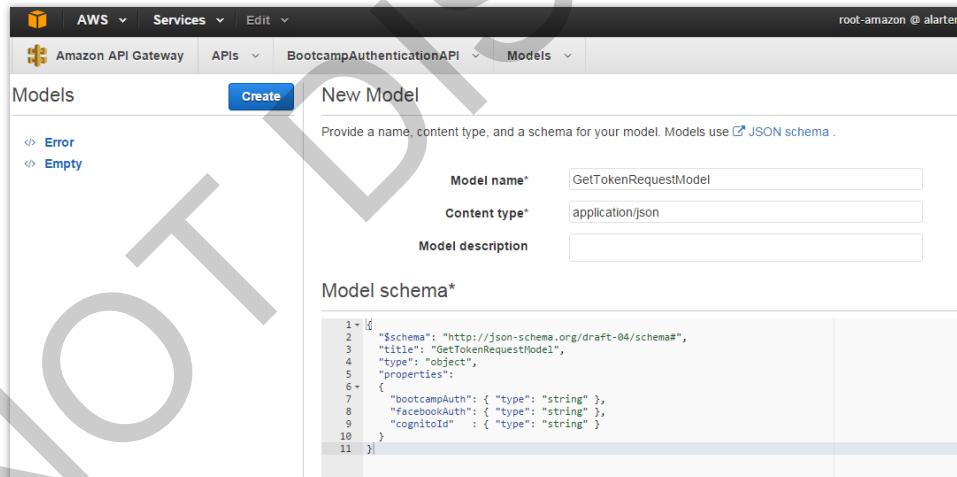
Note: It is critical that you enter the details exactly as shown below. If you alter the model in any way, our mobile application classes will not match the API interface, and your authentication process will not work as expected!

Model Name: GetTokenRequestModel

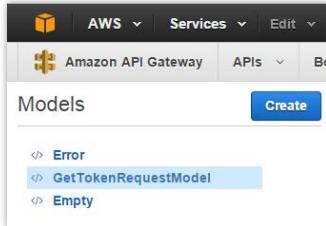
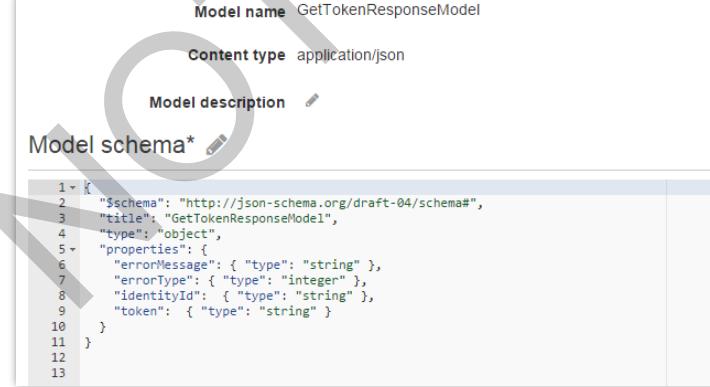
Content type: application/json

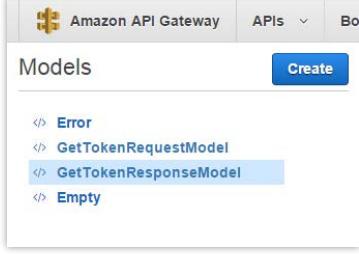
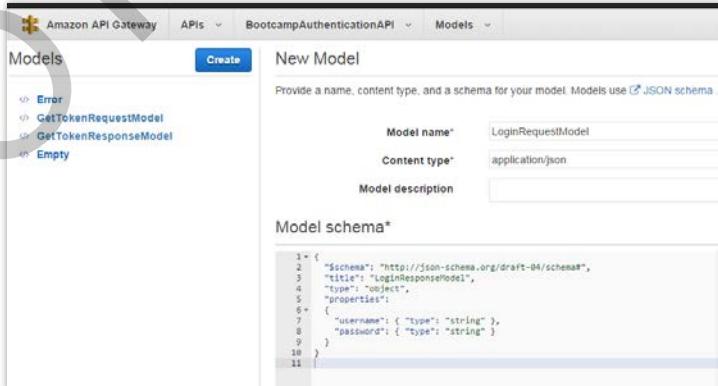
Model Schema:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "GetTokenRequestModel",  
    "type": "object",  
    "properties":  
    {  
        "bootcampAuth": { "type": "string" },  
        "facebookAuth": { "type": "string" },  
        "cognitoId" : { "type": "string" }  
    }  
}
```



Click the Create Model button at the lower right of the screen. You may need to scroll down to reveal it.

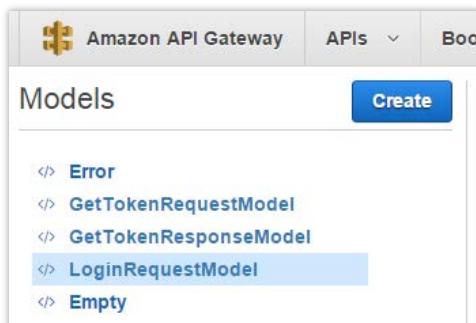
2.1.9	<p>GetTokenResponseModel</p> <p>Click the Create button in the top right in the navigation area.</p>  <p>As per the screen shot below, enter the following details.</p> <p>Note: It is critical that you enter the details exactly as shown below. If you alter the model in any way, our mobile application classes will not match the API interface, and your authentication process will not work as expected!</p> <p>Model Name: GetTokenResponseModel Content type: application/json Model Schema:</p> <pre>{ "\$schema": "http://json-schema.org/draft-04/schema#", "title": "GetTokenResponseModel", "type": "object", "properties": { "errorMessage": { "type": "string" }, "errorType": { "type": "integer" }, "identityId": { "type": "string" }, "token": { "type": "string" } } }</pre>  <p>2.1.10 Click the Create Model button in the lower right of the screen. You may need to scroll down to reveal it.</p>

2.1.11	<p>LoginRequestModel</p> <p>Click the Create button in the top right of the navigation area.</p>  <p>As per the screen shot below, enter the following details.</p> <p>Note: It is critical that you enter the details exactly as shown below. If you alter the model in any way, our mobile application classes will not match the API interface, and your authentication process will not work as expected!</p> <p>Model Name: LoginRequestModel Content type: application/json Model Schema:</p> <pre>{ "\$schema": "http://json-schema.org/draft-04/schema#", "title": "LoginRequestModel", "type": "object", "properties": { "username": { "type": "string" }, "password": { "type": "string" } } }</pre> 
2.1.12	<p>Click the Create Model button at the lower right of the screen. You may need to scroll down to reveal it.</p>

2.1.13

LoginResponseModel

Click the Create button in the top right in the navigation area.



As per the screen shot below, enter the following details.

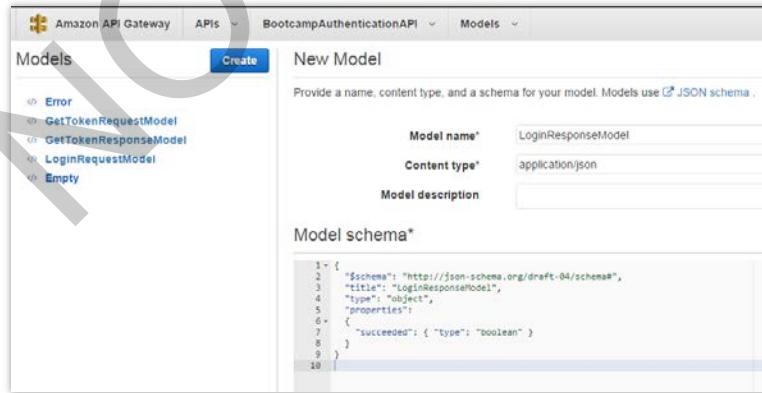
Note: It is critical that you enter the details exactly as shown below. If you alter the model in any way, our mobile application classes will not match the API interface, and your authentication process will not work as expected!

Model Name: LoginResponseModel

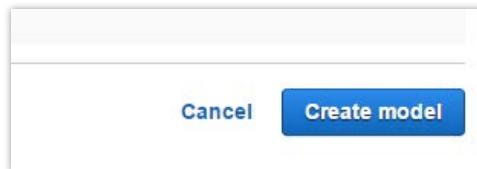
Content type: application/json

Model Schema:

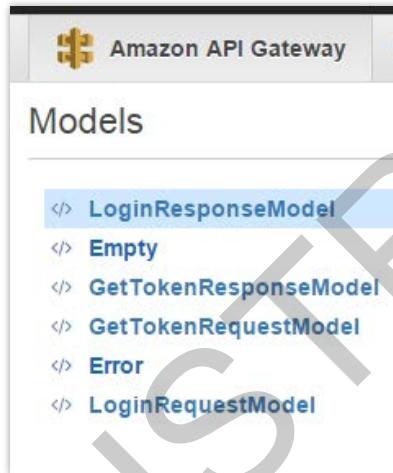
```
{  
  "$schema": "http://json-schema.org/draft-04/schema#",  
  "title": "LoginResponseModel",  
  "type": "object",  
  "properties":  
  {  
    "succeeded": { "type": "boolean" },  
    "message": { "type": "string" }  
  }  
}
```



- 2.1.14 Click the **Create Model** button at the lower right of the screen. You may need to scroll down to reveal it.

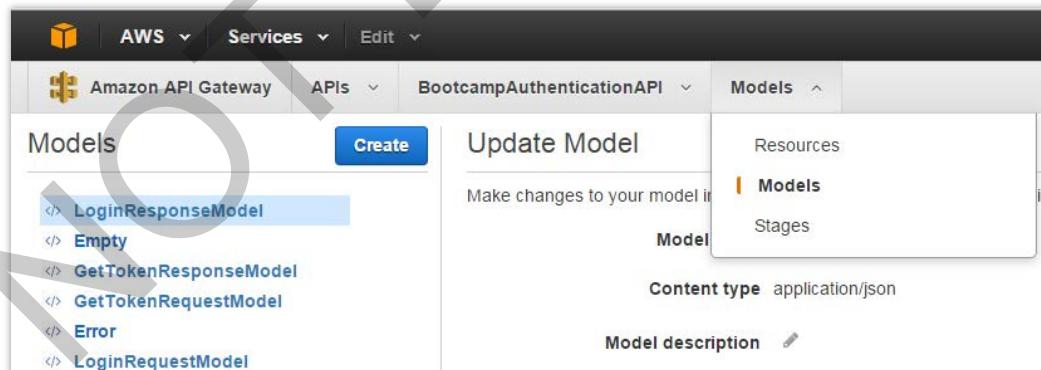


- 2.1.15 You have now created the four models we require for our Bootcamp authentication.

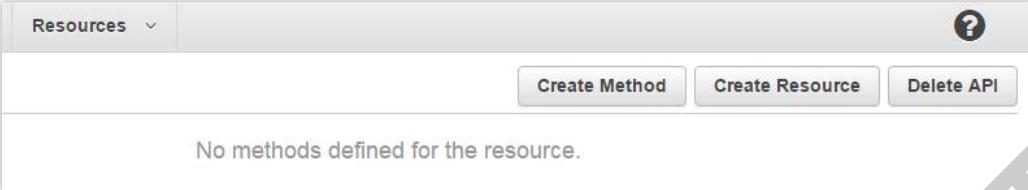
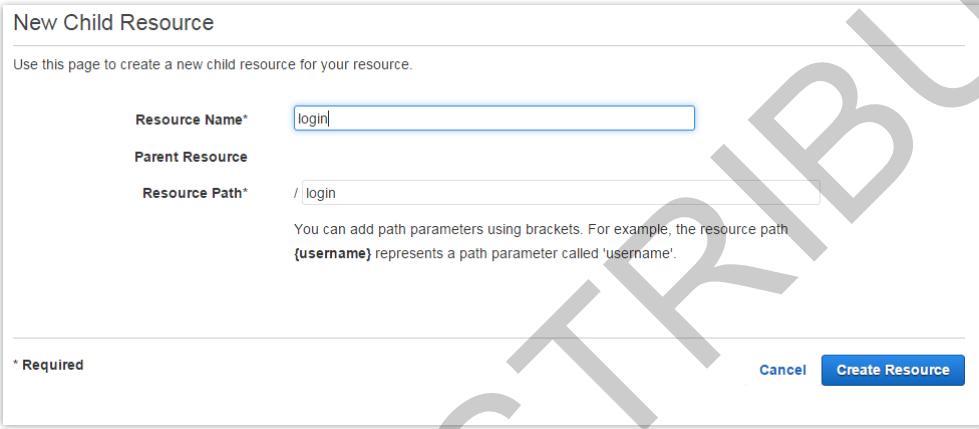
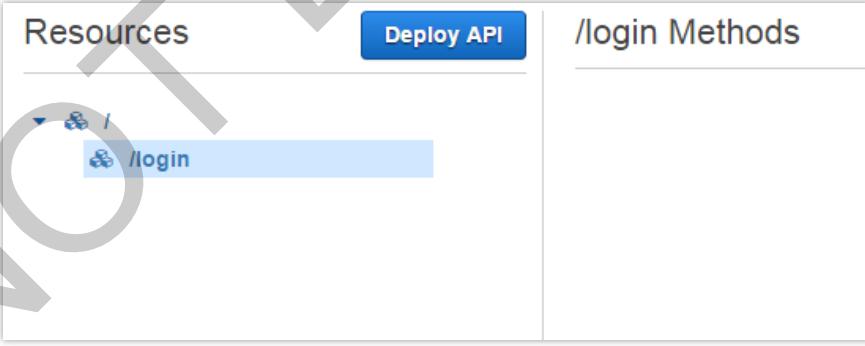


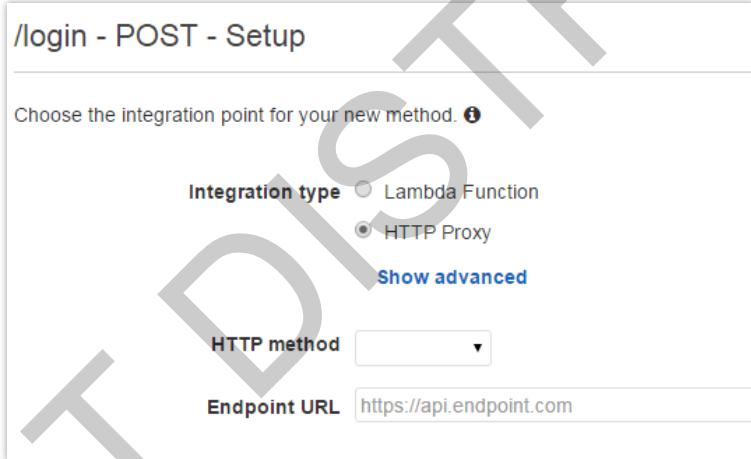
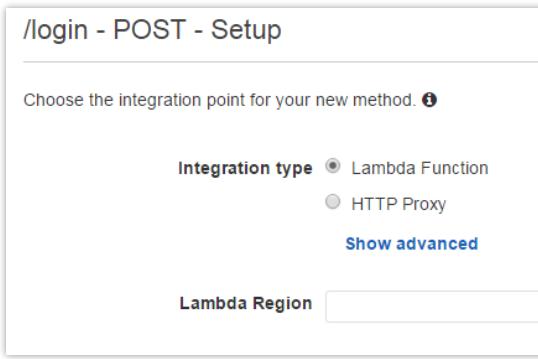
Next, we will create the Methods that we expose to our mobile application for our Bootcamp authentication.

In the drop-down list that currently says 'Models', select **Resources**.



As before, you will see a page that says there are no methods defined for the resource. We will now define new resources and methods for those resources.

2.1.16	<p>Click the Create Resource button in the top-right corner.</p> 
2.1.17	<p>Enter a name for your new resource. Enter login as the resource name.</p>  <p>Note: It is critical that you use the recommended name 'login' (without quotes) as the resource name in order for the mobile application to function correctly!</p>
2.1.18	<p>Click the Create Resource button. This will refresh the dashboard and show the new resource in the navigation pane on the left:</p> 
2.1.19	<p>Click the Create Method button in the top-right corner.</p>  <p>No methods defined for the resource.</p>

2.1.20	<p>In the left navigation pane, a new drop-down list will appear beneath the new ‘login’ resource we just created. From the drop-down list, select POST and click the grey circle with the tick icon next to the drop-down list to confirm:</p> 
2.1.21	<p>The page will now update and show the POST action in the left navigation pane in green to confirm that it has been created</p> <p>In the right pane, you will see the Setup page for your new method.</p> 
2.1.22	<p>Click the Lambda Function option from the radio list selection</p>  <p>When you do so, the pane will update itself to show Lambda-specific options which we will setup in the next steps</p>

2.1.23 In the **Lambda Region** drop-down list, choose the region you are running your Qwiklab in. Typically this will be **us-east-1**.

Then in the Lambda Function text box, enter

Bootcamp2015-DeveloperAuthentication

/login - POST - Setup

Choose the integration point for your new method. [?](#)

Integration type Lambda Function HTTP Proxy [Show advanced](#)

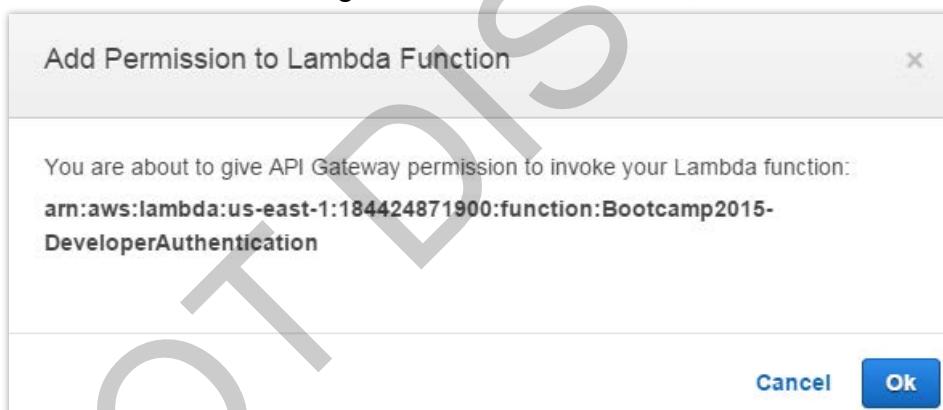
Lambda Region us-east-1

Lambda Function Bootcamp2015-DeveloperAuthentication

Save

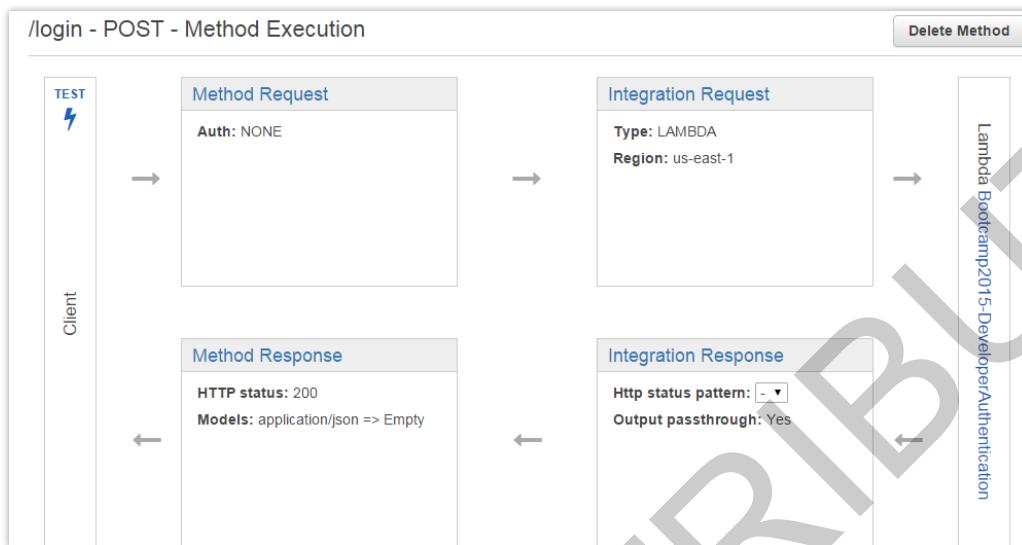
Click the **Save** button in the bottom-right corner to commit your changes.

2.1.24 You will see the following alert.



Click the **OK** button to grant permission

2.1.25 The right-hand panel will again update.



We will now set the Request model and Response model for our API calls

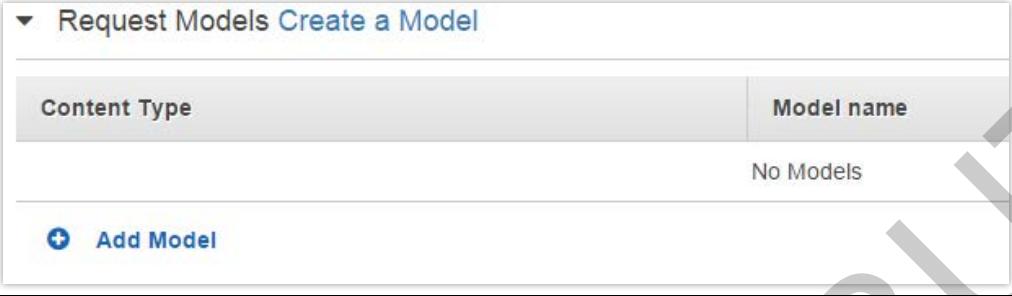
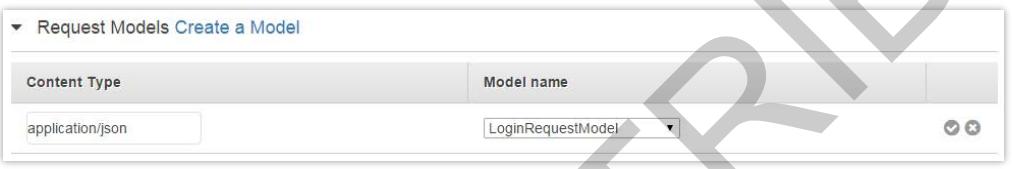
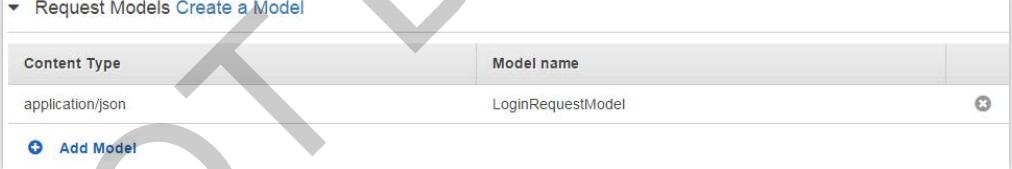
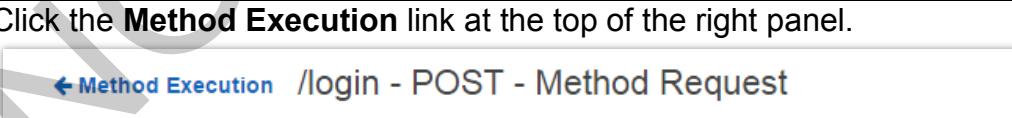
Click the title of the top-left box, labelled '**method request**'. The panel will scroll aside and update as shown:

The screenshot shows the "Method Execution /login - POST - Method Request" configuration panel. The "Authorization Settings" section is expanded, displaying:

- Authorization type**: NONE
- ARN**: arn:aws:execute-api:us-east-1:184424871900:aew97q6bsf/*/POST/login
- API Key Required**: false

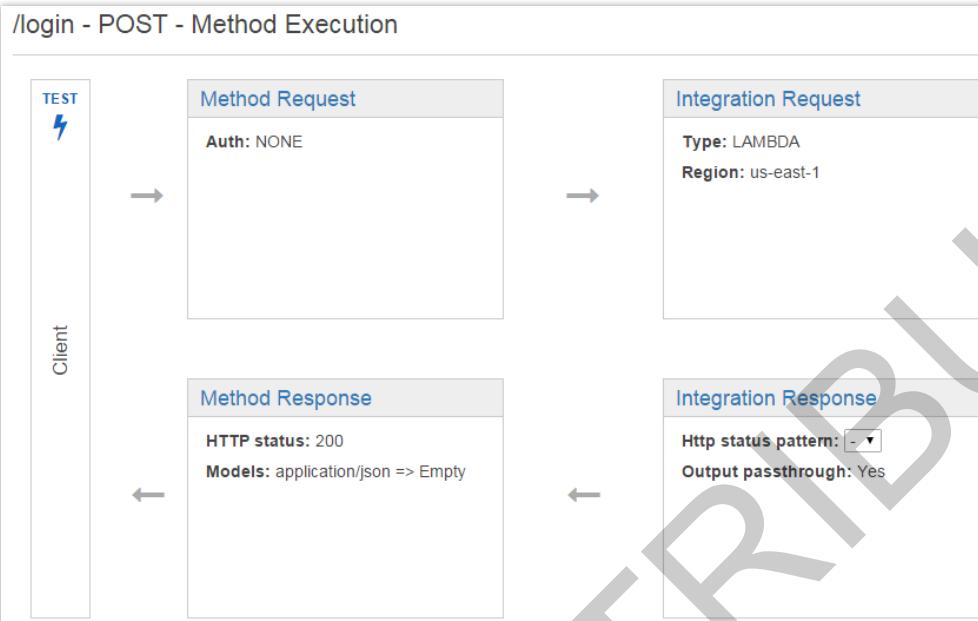
Below these settings, there are three expandable sections:

- URL Query String Parameters**
- HTTP Request Headers**
- Request Models** (with a "Create a Model" link)

2.1.26	<p>Click the Request Models twister (arrow at the left of the label) in the last row of the panel. This will reveal a new panel:</p> 
2.1.27	<p>Click the Add Model link:</p>  <p>In the Content Type text box, enter <code>application/json</code></p> <p>In the drop-down list under the Model Name column, choose LoginRequestModel.</p> <p>Click the grey circle with the tick icon, to confirm your changes. The panel will update as shown:</p> 
2.1.28	<p>Click the Method Execution link at the top of the right panel.</p>  <p>Provide information about this method's authorization settings and the parameters it can receive.</p> <p>Authorization Settings</p> <p>This takes the panel back to the main Method Execution section.</p>

2.1.29

Click the **Method Response** header link in the bottom-left box in this panel



This scrolls the panel to the Method Response page. Click the twister (the small triangle) at the left of the 200 under the Http Status section to reveal the method response details:

[← Method Execution /login - POST - Method Response](#)

Provide information about this method's response types, their headers and content types.

Http Status

▶ 200

[+ Add Response](#)

- 2.1.30 When you do this, the panel will expand:

The screenshot shows the 'Method Execution' interface for the '/login - POST' method. Under the 'Http Status' section, '200' is selected. The 'Response Headers for 200' section shows 'No headers'. The 'Response Models for 200' section shows 'Content type: application/json' and 'Models: Empty'. A 'Create a model' button is visible. Below these sections are 'Add Header' and 'Add Response Model' buttons.

- 2.1.31 Click the 'pencil' icon in the bottom-right corner of this panel, below the **Response Models for 200** section.

This enables the edit function for the Method Response model.

The screenshot shows the same configuration panel as above, but the 'Edit' mode has been activated for the 'Response Models for 200' section. The 'Content type' field now contains 'application/json' and the 'Models' dropdown menu is open, showing 'LoginResponseModel' as the selected option. A grey circle with a checkmark icon is visible next to the dropdown menu.

Choose **LoginResponseModel** from the drop-down list. Click the grey circle with the tick icon, to confirm your changes. The panel will update as shown.

The screenshot shows the configuration panel after the changes have been saved. The 'Content type' field still shows 'application/json' and the 'Models' dropdown now shows 'LoginResponseModel'. The grey circle with the checkmark icon is no longer visible, indicating the change has been confirmed.

2.1.32 Click the **Method Execution** link at the top of the right panel.

The screenshot shows the AWS API Gateway's Method Execution interface for the `/login - POST` method. At the top, there is a header with the link [Method Execution](#). Below it, a sub-header says `/login - POST - Method Response`. A note below the sub-header reads: "Provide information about this method's response types, their headers and content types." There is a table with one row labeled "Http Status" containing the value "200".

Below this, a large diagram illustrates the request-response flow:

- Client:** On the left, labeled "TEST" with a lightning bolt icon.
- Method Request:** Contains "Auth: NONE".
- Integration Request:** Contains "Type: LAMBDA" and "Region: us-east-1".
- Method Response:** Contains "HTTP status: 200" and "Models: application/json => LoginResponseModel".
- Integration Response:** Contains "Http status pattern: -" and "Output passthrough: Yes".

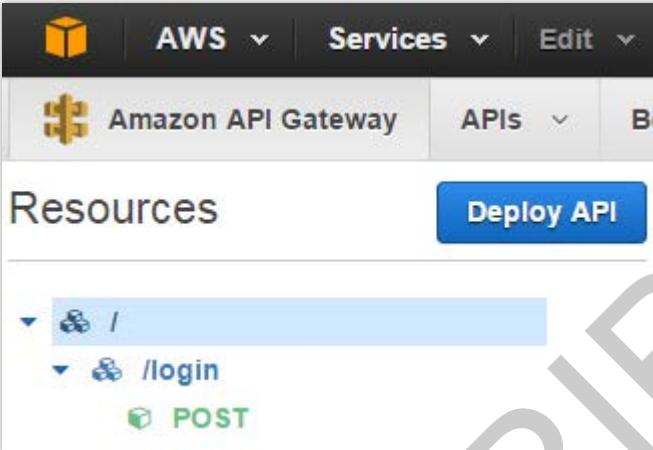
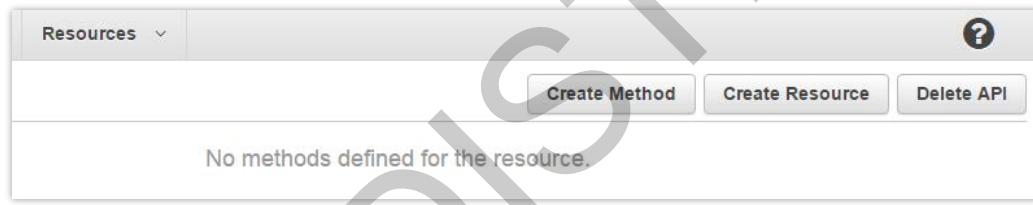
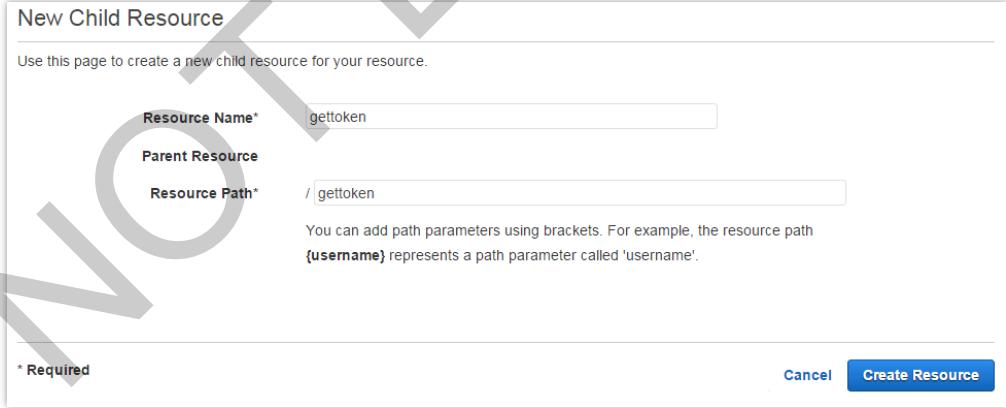
Arrows indicate the flow from Client to Method Request, Method Request to Integration Request, Integration Request to Method Response, and Method Response back to Client. The Integration Response section also has an arrow pointing towards the Method Response section.

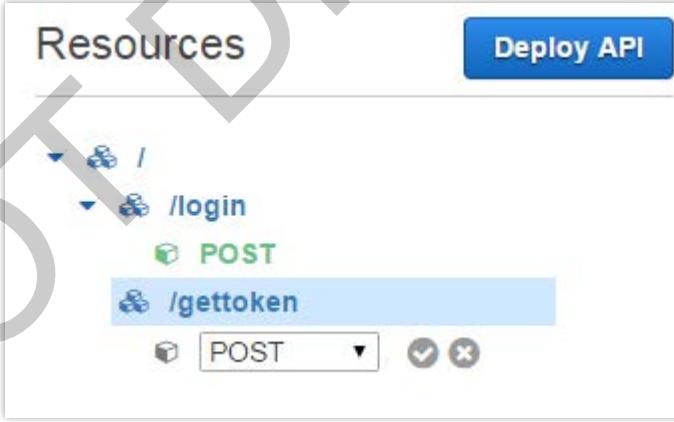
This takes the panel back to the main **Method Execution** section.

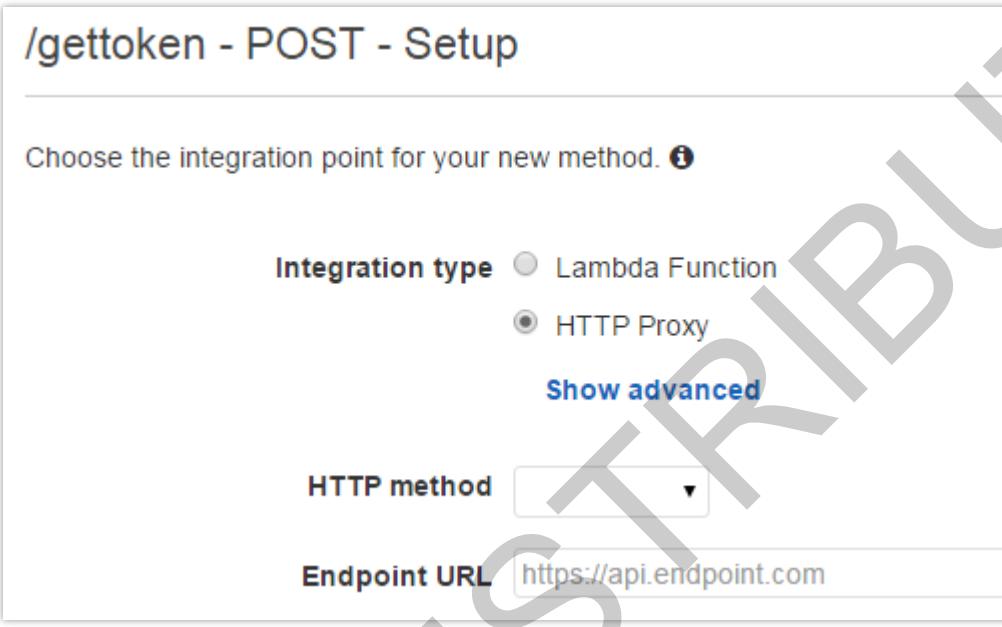
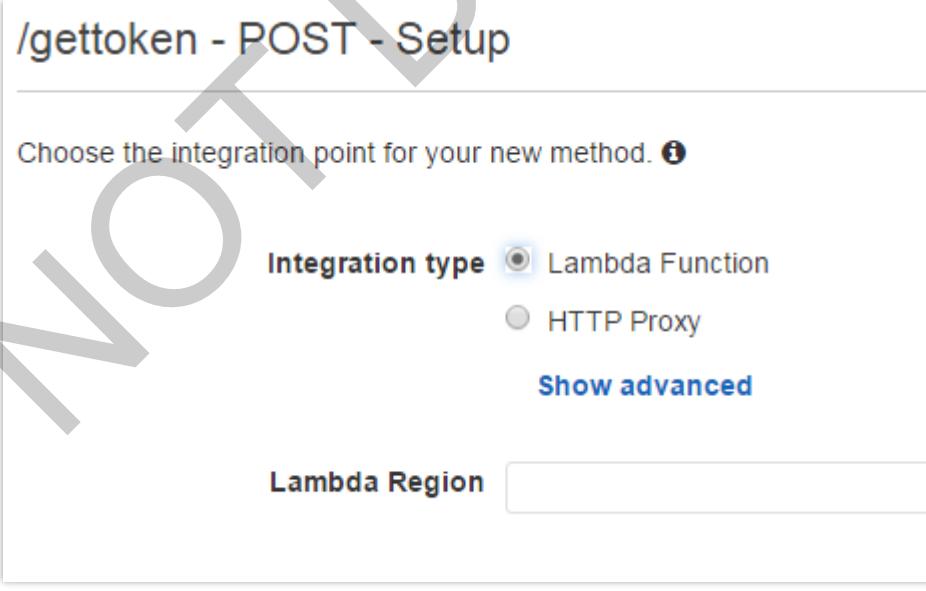
`/login - POST - Method Execution`

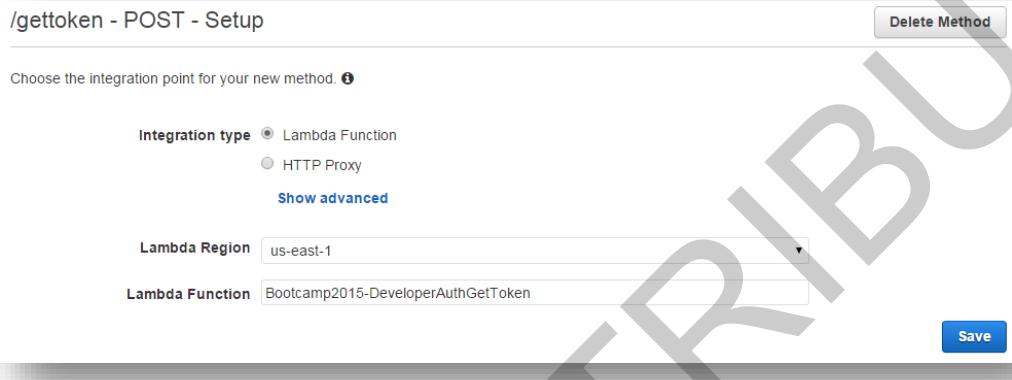
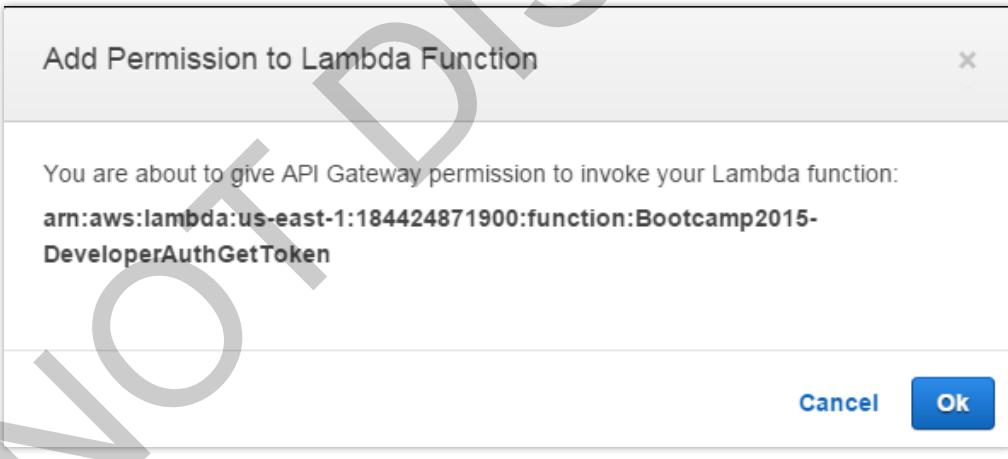
The *Bootcamp2015-DeveloperAuthentication* Lambda function is now configured, ready to be exposed via the API Gateway as `/login` from our root URL after we deploy our API.

We will now configure the *Bootcamp2015-DeveloperAuthGetToken* Lambda function via the API Gateway in the follow steps.

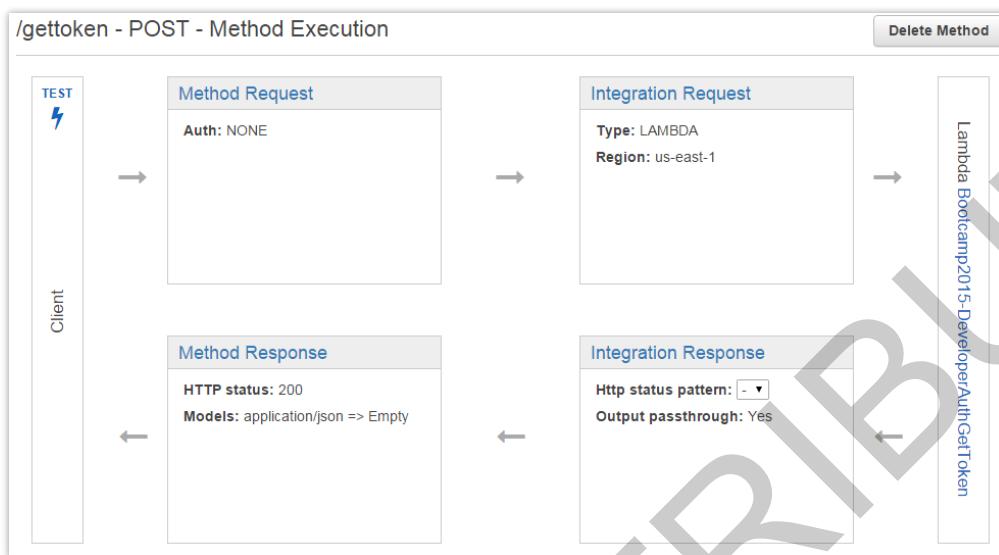
2.1.33	<p>In the left navigation panel, click the 'root' resource, which is indicated by the 'slash' at the top of the tree.</p>  <p>This will refresh the right panel.</p>
2.1.34	<p>Click the Create Resource button in the top-right corner, in the right panel:</p> 
2.1.35	<p>Name your new resource gettoken</p>  <p>Note: It is critical that you use the recommended name 'gettoken' (without quotes) as the resource name in order for the mobile application to function correctly!</p>

2.1.36	<p>Click the Create Resource button. This will refresh the dashboard and show the new resource in the navigation pane on the left:</p> 
2.1.37	<p>Click the Create Method button in the top-right corner.</p> 
2.1.38	<p>On the left navigation pane, a new drop-down list will appear below the new 'gettoken' resource we just created. From the drop-down list, select POST and click the grey circle with the tick icon next to the drop-down list to confirm:</p> 

2.1.39	<p>The page will now update and show the POST action in green in the left navigation pane to confirm that it has been created.</p> <p>In the right pane, you will see the Setup page for your new method.</p> 
2.1.40	<p>Click the Lambda Function option from the radio list selection.</p>  <p>When you do so, the pane will update itself to show Lambda-specific options which we will setup in the next steps.</p>

2.1.41	<p>In the Lambda Region drop-down list, choose the region you are running your Qwiklab in. Typically this will be us-east-1.</p> <p>Then, in the Lambda Function text box, enter</p> <p style="text-align: center;">Bootcamp2015-DeveloperAuthGetToken</p>  <p>Click the Save button in the bottom-right corner to commit your changes.</p>
2.1.42	<p>You will see the following alert.</p>  <p>Click the OK button to grant permission.</p>

2.1.43 The right panel will again update.



We will now set the Request model and Response model for our API calls.

Click the **Method Request** box in the top-left corner. The panel will scroll aside and update as shown.

Method Execution /gettOKEN - POST - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Authorization Settings

Authorization type: NONE

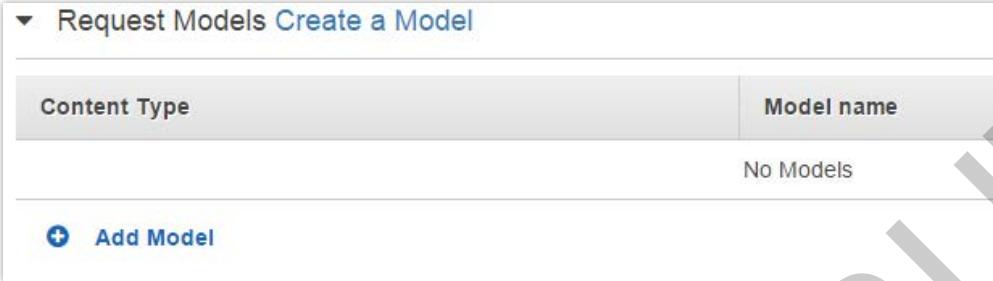
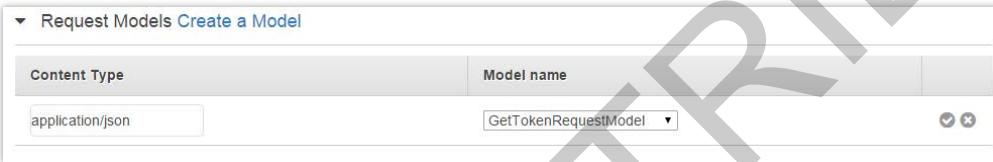
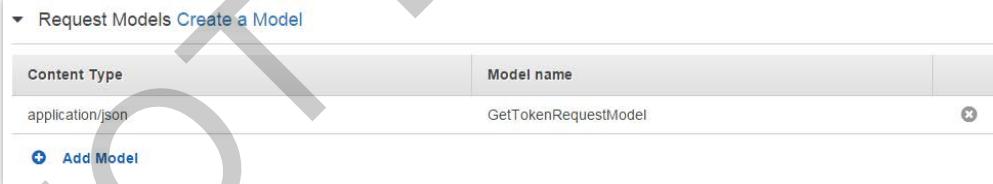
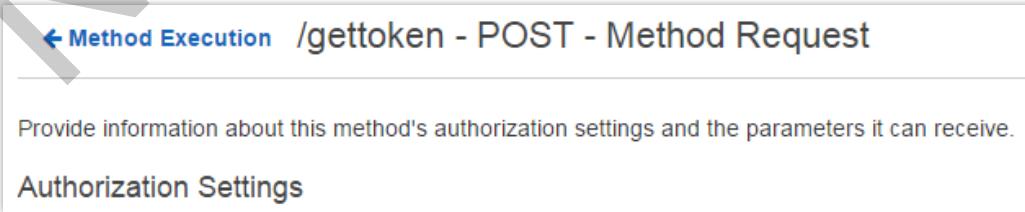
ARN: arn:aws:execute-api:us-east-1:184424871900:aew97q6bsf/*POST/gettOKEN

API Key Required: false

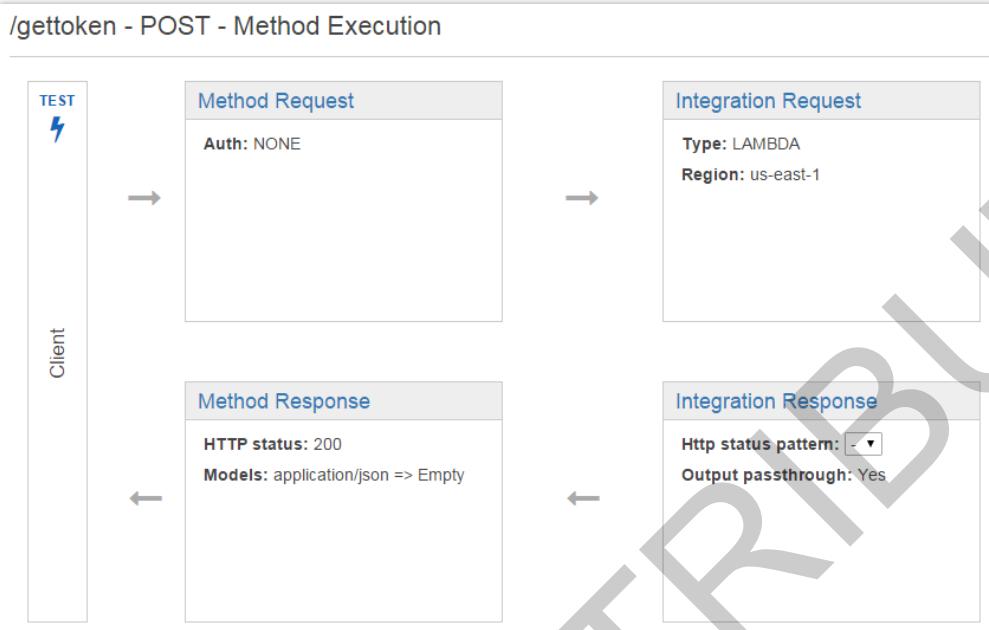
URL Query String Parameters

HTTP Request Headers

Request Models [Create a Model](#)

2.1.44	<p>Click the Request Models twister (arrow at the left of the label) in the last row of the panel. This will reveal a new panel:</p> 
2.1.45	<p>Click the Add Model link.</p>  <p>In the Content Type text box, enter <code>application/json</code></p> <p>In the drop-down list under the Model Name column, choose the GetTokenRequestModel.</p> <p>Click the grey circle with the tick icon, to confirm your changes. The panel will update as shown.</p> 
2.1.46	<p>Click the Method Execution link at the top of the right panel.</p>  <p>Provide information about this method's authorization settings and the parameters it can receive.</p> <p>Authorization Settings</p> <p>This takes the panel back to the main Method Execution section.</p>

- 2.1.47 Click the **Method Response** header link in the bottom-left box in this panel



This scrolls the panel to the Method Response page.

← Method Execution /gettoken - POST - Method Response

Provide information about this method's response types, their headers and content types.

Http Status

▶ 200

+ Add Response

- 2.1.48 Click the 'twister', the arrow on the left of the Http Status 200. This reveals other options.

Http Status	
▼	200
Response Headers for 200	
Name	
No headers	
+ Add Header	
Response Models for 200	
Content type	Models
application/json	Empty
+ Add Response Model	

- 2.1.49 Click the ‘pencil’ icon at the bottom-right of this panel. This enables the edit function for the Method Response model.

Response Headers for 200

Name

No headers

Response Models for 200

Content type

Models

application/json

GetTokenResponseModel

Create a model

Choose **GetTokenResponseModel** from the drop-down list. Click the grey circle with the tick icon, to confirm your changes. The panel will update as shown:

Method Execution /gettoken - POST - Method Response

Delete Method

Provide information about this method's response types, their headers and content types.

Http Status

200

Response Headers for 200

Name

No headers

Add Header

Response Models for 200

Content type

Models

application/json

GetTokenResponseModel

Add Response Model

Create a model

2.1.50 Click the **Method Execution** link at the top of the right panel.

This takes the panel back to the main **Method Execution** section.

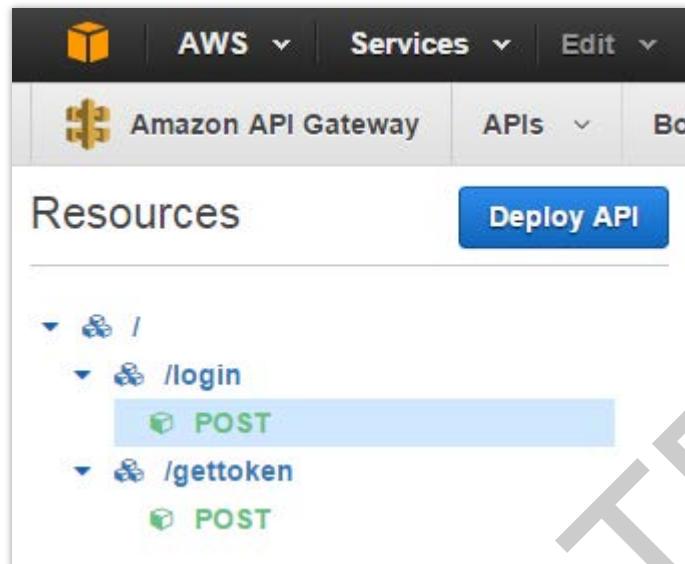
The *Bootcamp2015-DeveloperAuthGetToken* Lambda function is now configured, ready to be exposed via the API Gateway as /gettoken from our root URL after we deploy our API.

We are now ready to test our API before we make it available for the mobile app.

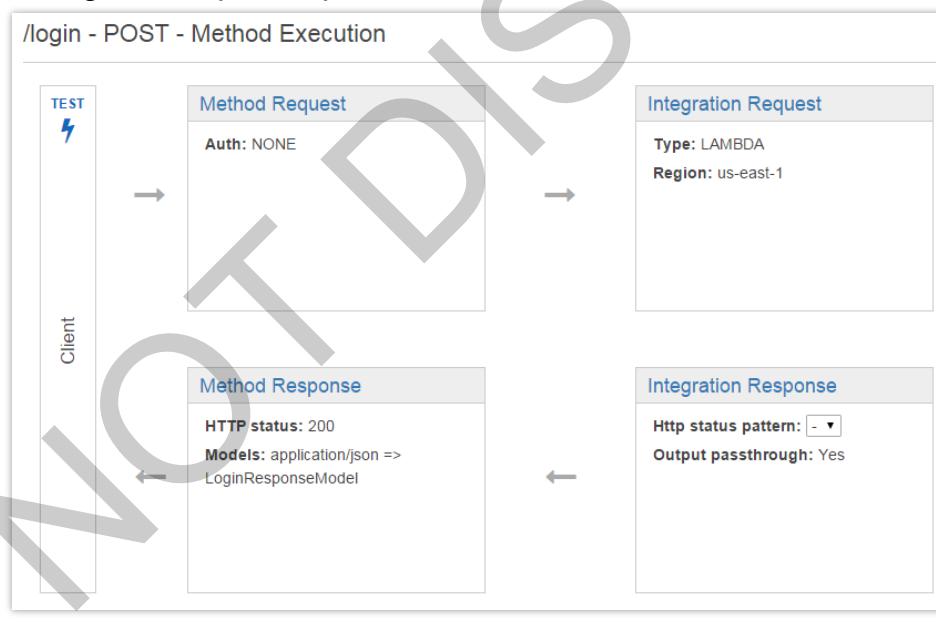
2.1.51

Testing the /login method:

In the navigation panel on the left, click the POST child entry of the /login resource.



The right-hand panel updates as shown here:



2.1.52	<p>Click the Test link icon on the left.</p>  <p>The panel updates as shown here:</p> <div data-bbox="326 528 1321 802"><p>◀ Method Execution /login - POST - Method Test</p><p>Make a test call to your POST method. Provide the value of the input parameters for your API call.</p><p> POST » /login</p><p>▶ Request Body</p></div> <p>Click the ‘twister’ (small arrow) next to the Request Body heading, to reveal the Request Body text entry field:</p> <div data-bbox="326 929 1387 1647"><p>▼ Request Body</p><div data-bbox="375 1013 1305 1626"><p>1</p></div></div>
--------	--

- 2.1.53 In the Request Body text entry field, enter the following text:

```
{  
    "username": "username", "password": "password"  
}
```

This simulates the payload that the mobile app will send through when it wants to authenticate the user.

← Method Execution /login - POST - Method Test

Make a test call to your **POST** method. Provide the value of the input parameters for your API call.

POST » /login

▼ Request Body

```
1 [ ]  
2 "username": "username", "password": "password"  
3 }
```

Click the Test button, in the bottom-right corner (you may have to scroll to see it).



2.1.54 The function will execute. You should see the following output:

Request: /login

Status: 200

Latency: 1202 ms

▼ Response Body

```
{  
  "succeeded": true,  
  "message": "OK"  
}
```

► Response Headers

► Logs

If you see

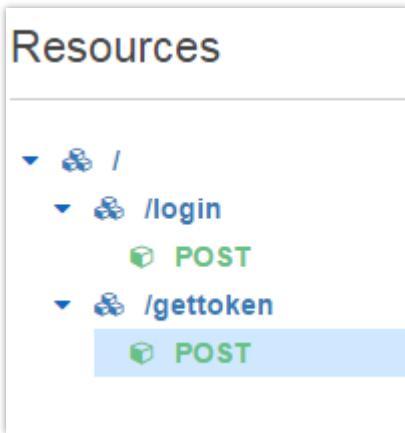
```
{  
  "succeeded": true,  
  "message": "OK"  
}
```

then your endpoint for login is working as expected! If not, check your steps from above.

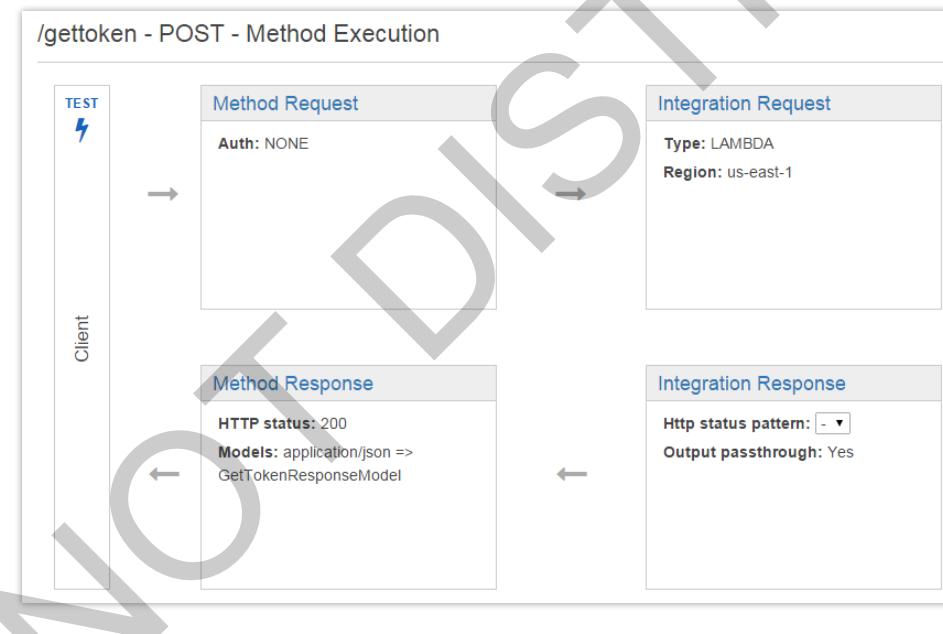
2.1.55

Testing the /gettoken method:

In the left-hand navigation panel, click the POST child of the **/gettoken** resource.



The right-hand panel updates as shown here:



- 2.1.56 Click the **Test** link on the left.



The panel updates as shown here:

◀ Method Execution /gettoken - POST - Method Test

Make a test call to your **POST** method. Provide the value of the input parameters for your API call.

cube **POST** » /gettoken

▶ Request Body

Click the 'twister' (small arrow) next to the Request Body heading, to reveal the Request Body text entry field.

▼ Request Body

1

- 2.1.57 In the Request Body text entry field, enter the following text:

```
{  
    "bootcampAuth" : "TEST",  
    "cognitoId"     : "us-east-1:aaaaaaaa-bbbb-cccc-dddd-  
aaaaaaaa"  
}
```

NOTE: You should replace the cognitoid above with an identity from your own Cognito Identity Pool's Identity Browser.

This simulates the payload that the mobile app will send through when it wants to refresh the authentication token for the user.

◀ Method Execution /gettoken - POST - Method Test

Make a test call to your **POST** method. Provide the value of the input parameters for your API call.

POST » /gettoken

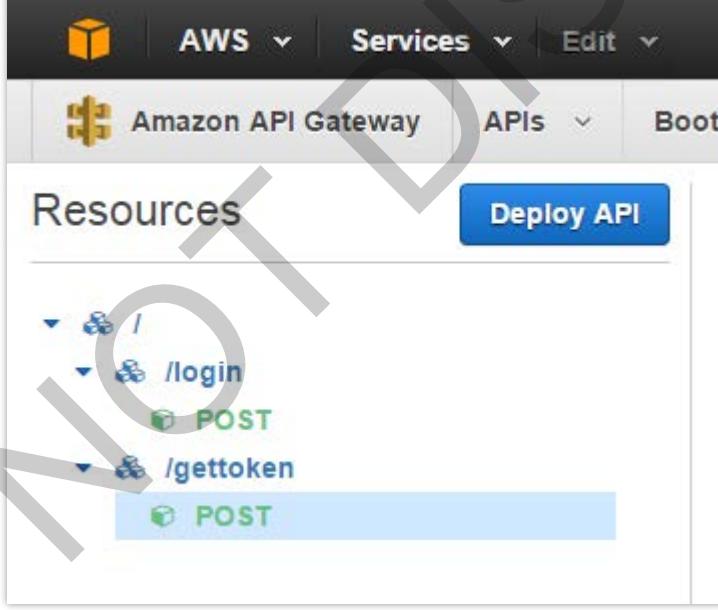
▼ Request Body

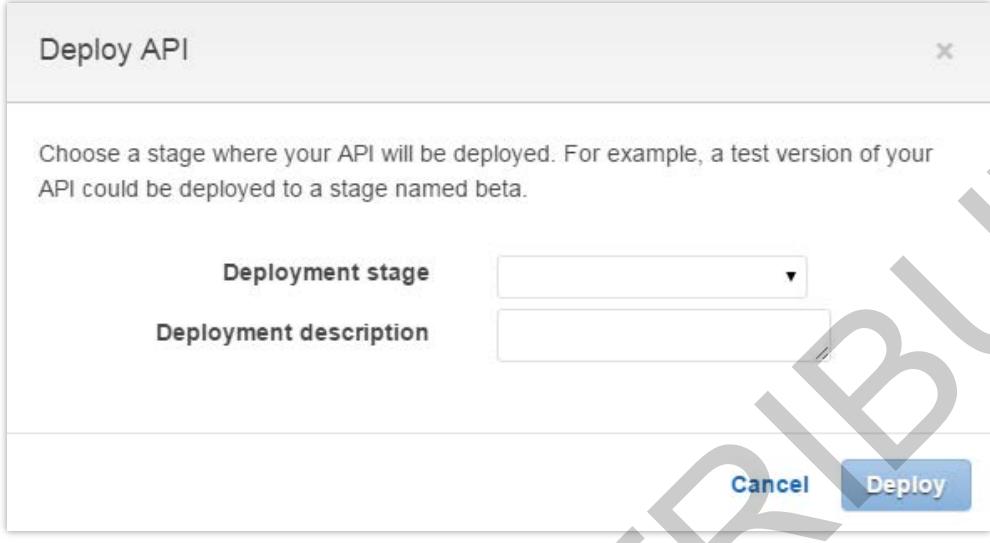
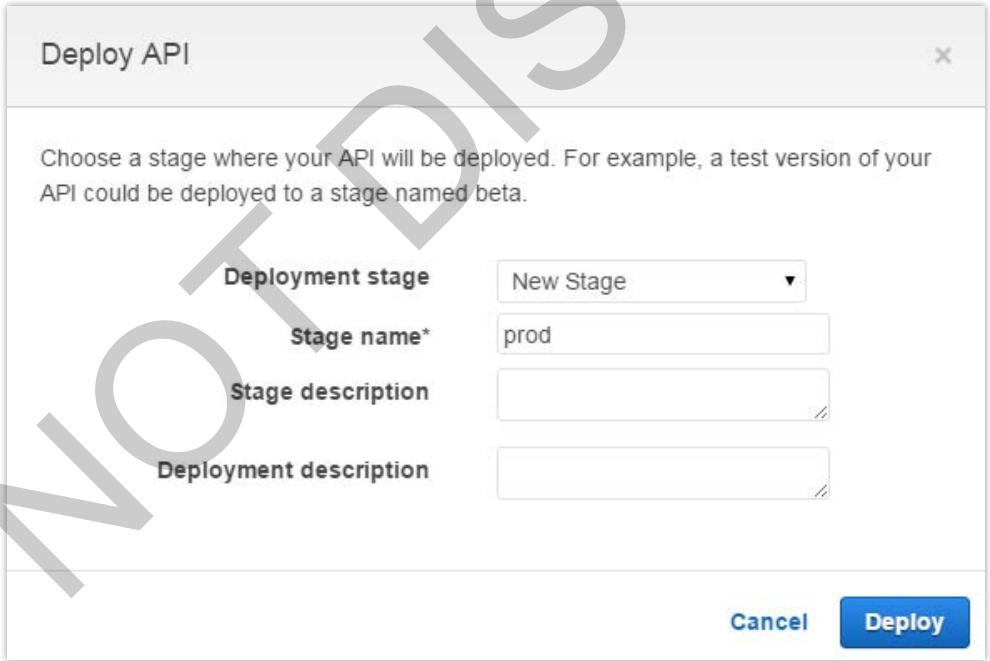
```
1 ▾ {  
2     "bootcampAuth" : "TEST",  
3     "cognitoId"     : "us-east-1:aaaaaaaa-bbbb-cccc-dddd-  
        -aaaaaaaa"  
4 }  
5
```

Click the Test button in the bottom-right corner (you may have to scroll down to see it).



2.1.58	<p>The function will execute. You should see the following output:</p> <pre>Request: /gettoken Status: 200 Latency: 242 ms ▼ Response Body { "errorMessage": "Identity 'us-east-1:aaaaaaaa-bbbb-cccc-dddd-aaaaaaaa' not found.", "errorType": 1, "identityId": "", "token": "" } ▼ Response Headers ▼ Logs</pre> <p>Of course, you have provided a dummy Cognito ID to identify your user, and so Cognito has failed to generate the token correctly. However, if your API Gateway managed to call your Lambda function, and it managed to Cognito to attempt to generate your token, your configuration is correct and working as expected!</p>
2.1.59	<p>If you provide a real Cognito Identity from your Cognito Identity Pool, you will see something like the following:</p> <pre>Request: /gettoken Status: 200 Latency: 217 ms ▼ Response Body { "errorMessage": "", "errorType": 0, "identityId": "us-east-1:9171fb7f-591e-42c7-9931-a32ebf31d192", "token": "eyJraWQiOiJ1cy1lYXN0LTExIiwidHlwIjoiSldTIiwiYnxnIjo1UlM1MTIifQ.eyJzEtyTMyZWJmMzFkMTkyIiwiYXVkJioidXMtZWfdC0xOjRhMmVhM2I2LTUwNmYtNDdmNS04ODFkLTFnNwLmN1c3RvbS5ib290Y2ftcDIwlMTUiXSviaXNzIjoiaHR0cHM6Ly9jb2duaXRvLwIkZW50aXR5LmfTYxEOTQzMjczMX0.WJ4d6-bu6hJAjf-Aoi39I-bsdbroChxFeyuoyyqtpn-wrOPjPpThxNht5skYE_Qj3vxY_oeoqDXPP5DyoCQxB2zfWYFUKoWlV18FLbuczGUfdnmOxMbV0cnsCfBf0coY7FZL6md2bmz5Pji-hJqzwKaGe5EFMd_vLos5MuDgeM9_5Sd_IxjNiLls_7o0dqnsT8v005CN6W615whmJcElzW4-w05wAGj" }</pre> <p>This indicates that the API call has successfully called into Cognito using the AWS SDK to generate a token for your user. Note that the errorType value is zero, i.e. there are no errors.</p>

2.1.60	<p>Possible failure scenario:</p> <p>If you see</p> <pre>{ "errorMessage": "You forgot to configure your identity pool id in the Lambda function!", "errorType": 3, "identityId": "", "token": "" }</pre> <p>then it means you forgot to update the Lambda function to provide your valid Cognito Identity Pool Id in step 1.2.10 above. Update your Lambda function as per the details in the task step to fix this problem.</p>
2.1.61	<p>If your tests above are successful as per the instructions, you are ready to 'deploy' your API to make it ready for use by your mobile application.</p> <p>In the Navigation pane on the left, locate the Deploy API button and click it.</p> 

2.1.62	<p>The Deploy API dialog will appear.</p> 
2.1.63	<p>Because you have not deployed the API before, there will be no Stages available so we need to create one. We can do this by selecting 'New Stage' from the drop-down list.</p>  <p>Enter prod as the name for the new stage.</p> <p>Click the Deploy button.</p>

- 2.1.64 The **prod Stage Editor** will appear when the deployment is complete.

The screenshot shows the 'prod Stage Editor' configuration page. At the top, there is a blue header bar with the text 'Invoke URL: https://aew97q6bsf.execute-api.us-east-1.amazonaws.com/prod'. Below the header, there are three tabs: 'Settings' (which is selected), 'SDK Generation', and 'Deployment History'. The main content area is titled 'Configure the metering and caching settings for the prod stage.' It contains three sections: 'Cache Settings' (with an 'Enable API cache' checkbox), 'CloudWatch Settings' (with 'Enable CloudWatch Logs' and 'Enable CloudWatch Metrics' checkboxes), and 'Throttling Settings' (with 'Burst Limit' and 'Rate' input fields). The entire interface has a light gray background with white text and blue highlights for the active tab.

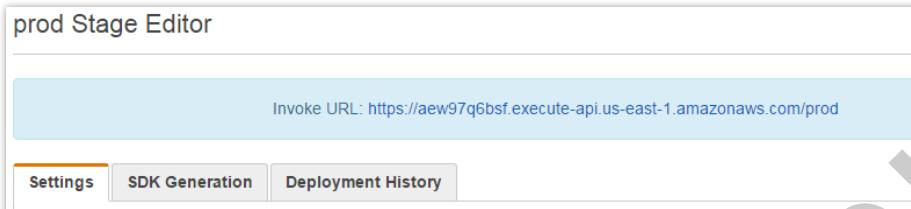
For our Bootcamp, we will not take advantage of the additional features of the API Gateway. As part of the Bootcamp Mobile Application code, your instructors have already downloaded and included the SDK for your mobile platform, so you don't have to do that during the Bootcamp.

Optional, if time permits

If you are interested in seeing what the SDK generator will create for you, feel free to download the relevant ZIP file for your mobile platform.

2.1.65 You now need to update the **bootcamp-config.js** file on your Edison device to include the new API Gateway endpoint URL.

The **Invoke URL** is shown in the blue section at the top of the panel, for example:



Note that your URL will be different.

Copy the URL, protocol (HTTPS, in the above example) included. So for the above example, you would copy

<https://aew97q6bsf.execute-api.us-east-1.amazonaws.com/prod>

into your clipboard. You need to paste this into your **bootcamp-config.js** file on your Edison device.

In an SSH session to your Edison, type in the following command:

```
vi /etc/bootcamp2015-config.js
```

Locate the setting for the API Gateway URL, which is called

API_GATEWAY_ENDPOINT_URL

```
//////////  
//  
// Configure these values  
//  
//////////  
  
var DEVICE_NAME = "Mini Edison";  
var AWS_ACCOUNT_ID = "0000000000";  
var COGNITO_IDENTITY_POOL_ID = "us-east-1:XXXXXXXXXXXX";  
var MOBILE ANALYTICS APP ID = "0000000000";  
var API GATEWAY ENDPOINT URL = "https://REPLACE THIS WITH API GATEWAY URL FROM LAB GUIDE";  
var APPLIANCE_TYPE = 5;  
var BLE_UUID = "badf00d5dff48d2b060d0f5a71096e0";  
var GOOGLE_PROJECT_NUMBER = "0000000000";
```

The default configuration indicates that you need to replace the URL with the API Gateway URL from the lab guide – you will do this next.

2.1.66	<p>When you have located the ‘dummy’ URL in the config, erase it and replace it with your API Gateway URL, which you have in your clipboard.</p> <p>You can do this by positioning the cursor (using the cursor keys) on the ‘h’ of the ‘https:// ..’ value, and pressing the ‘x’ key on the keyboard until all the characters in the dummy value have been erased, leaving only the value “”.</p> <p>Then, move the cursor between the two quotes, press the i key (to go into insert mode) and right-click on your mouse. This will paste the value you have in your clipboard, which is the URL of your new endpoint for the API Gateway.</p> <p>When you have the value pasted in, press Esc then enter</p> <p>:x</p> <p>to save the file.</p>
2.1.67	<p>You now need to have your Edison re-register itself to the Central web site, so that the new API Gateway URL is recorded, and can be picked up by the mobile app</p> <p>On your Edison SSH session, enter the following command:</p> <pre>systemctl restart bootcamp-register.service</pre> <p>The updated URL will now be recorded on the Central web site.</p> <p>In the next section of the Bootcamp, we make use of the API Gateway and Lambda functions we have just created to provide Developer Authentication for our Mobile application.</p>

Lab 8: Building and Running the Companion Mobile App and Implementing the Appliance Control Panels

Overview

In this lab, we will build and test the Companion Mobile Application that provides a user experience for the customers of our Connected Domestic Appliance Network.

Most of the code is already in place, leaving you to concentrate on understanding and implementing the Control Panels, which make use of Cognito Sync to share data between devices and platforms

This Lab is not intended to teach you how to be an Android/iOS programmer, but instead, assumes you have experience with these platforms and focuses on how to use the AWS SDKs to integrate your applications and take advantage of the AWS Mobile Optimized Services and Connectors

Before commencing this Lab guide, you must have previously installed Android Studio or XCode depending on what platform you are targeting – see the Bootcamp prerequisites documentation

Objectives	After completing this lab, you will be able to: <ul style="list-style-type: none">• Create SNS Platform Applications for Cognito Sync and general application push notifications• Enable Cognito Push Sync for our Identity Pool• Set up your development environment, to build and run the Bootcamp Companion Mobile Application• Test out the application's features – Tethering, Developer Authentication, Facebook Authentication, Pairing, Chat and Control Panels
Pre-requisites	This lab requires: <ul style="list-style-type: none">• Setup of either the Google Developer project or Apple Developer project following the prerequisites guide• Completion of Lab Guide 3 task 2.1.15 where the Google Project Number was added to the /etc/bootcamp-config.js file on the Edison• Android Studio or XCode depending on whether you intend to target Mac/iOS or Windows/Android for the development environment and mobile device• Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">◦ Note The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.• For Microsoft Windows users: Administrator access to the computer.• An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).
Duration	45 minutes

Important Note

Prior to attending the Bootcamp, we requested that you set up your Google Developer Account or Apple Developer Account and create the relevant resources such as a new Google Project using the Google developer console, or a new App and Provisioning Profile in the Apple Developer Member Center.

If you haven't completed these tasks, you will not be able to complete the remaining sections of this lab guide.

You should stop here, and complete the prerequisites then come back to this point.

Task 1: Creating New Platform Applications for Push Synchronization and Messaging

Task 1-1: Creating New Platform Applications for Push Synchronization—for Android

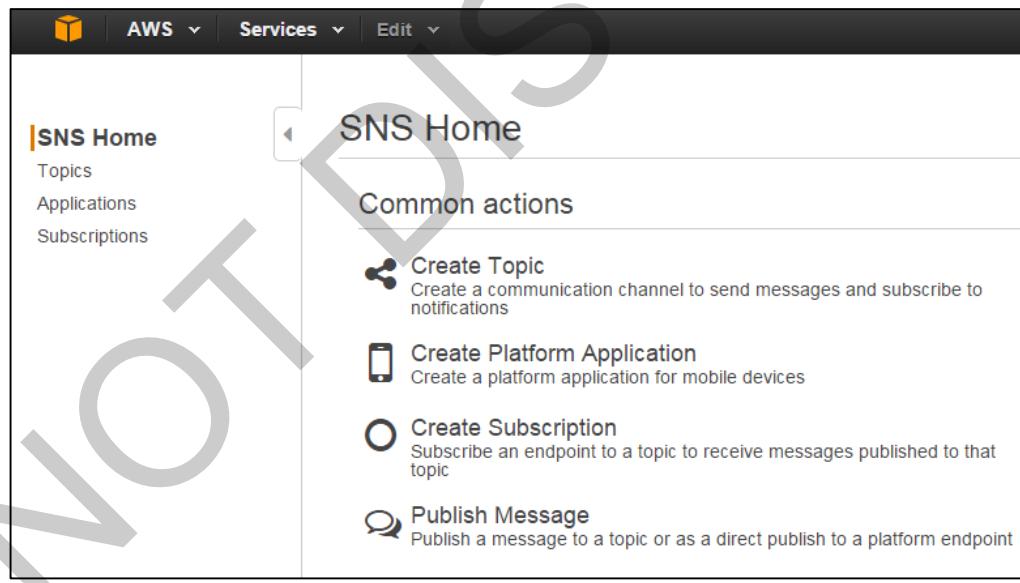
Overview	<p>In this section we will create a new SNS Platform Application that will be used for Cognito Push Sync with Android.</p> <p><i>If you are not using Android, you should skip this section and instead locate the section for iOS SNS mobile push, in this same lab guide</i></p>
-----------------	--

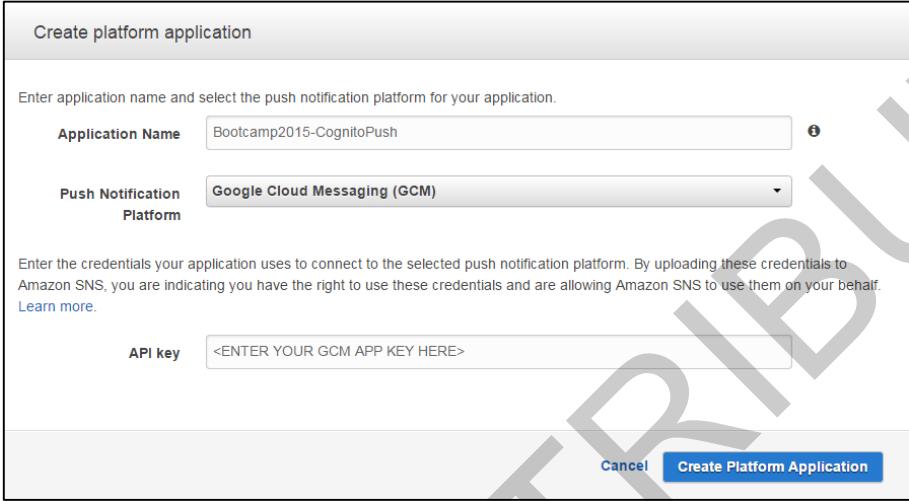
Step	Instruction
------	-------------

- 1.3.1 Login to the AWS Management Console from your Qwiklabs Environment and choose **Services**→**Mobile Services**→**SNS**



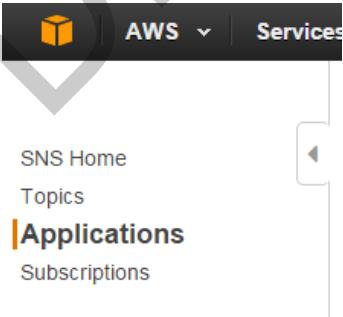
- 1.3.2 In the AWS console for SNS, click the **Create platform application** link in the main panel.

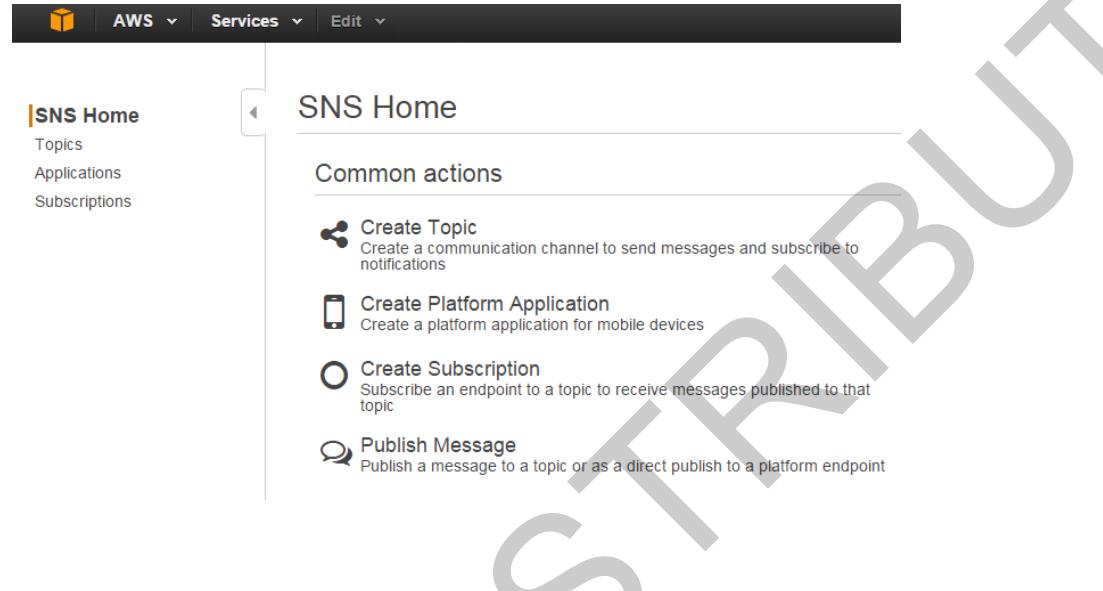
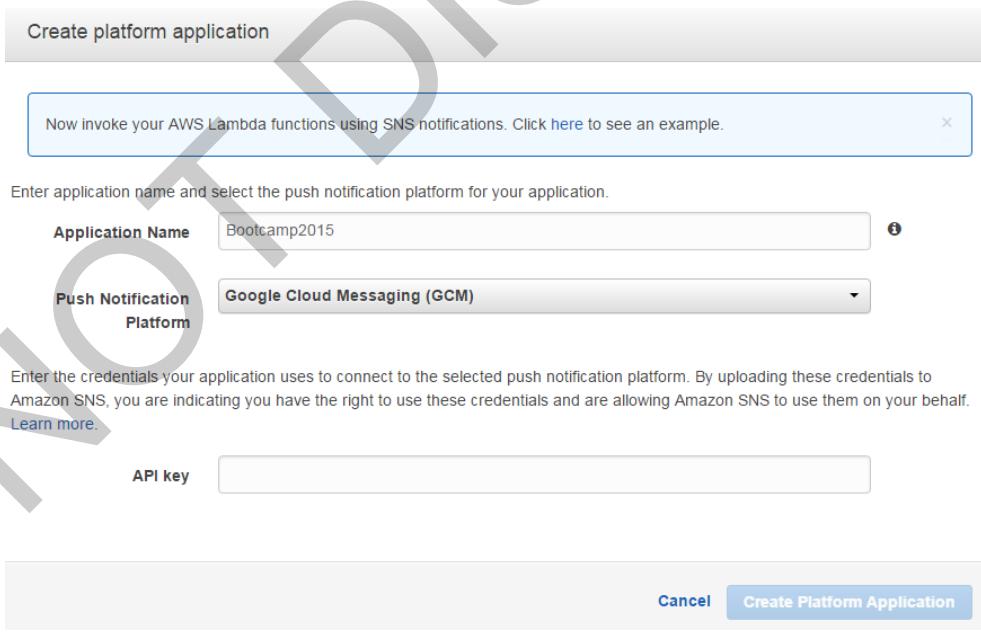


1.3.3	<p>Name the platform application Bootcamp2015-CognitoPush</p>  <p>Select the Push Notification platform. For Android, we must use Google Cloud Messaging (GCM).</p>			
1.3.4	<p>Enter the API key from your Google console. If you don't have the API key handy, you can get the API key from your Google Developer Console by browsing to https://console.developers.google.com/project</p> <p><i>Note: As part of the prerequisites, we requested that you create a Google Project prior to attending the Bootcamp. If you have not done this, you will need to pause here, and follow the steps in the prerequisites information</i></p> <p>Paste your API Key, which will look something like this:</p> <p>AlzaSyDdF-3XXxXXsXxxxxjNGzkx1GHb3xxUBJU</p> <p>Click Create Platform Application</p>			
1.3.5	<p>Application Details: Bootcamp2015-CognitoPush (GCM)</p> <table border="1" data-bbox="323 1503 1155 1691"> <tr> <td>Platform Application Actions ▾</td> </tr> <tr> <td>Application ARN arn:aws:sns:us-east-1:406334898880:app/GCM/Bootcamp2015-CognitoPush</td> </tr> <tr> <td>Platform GCM</td> </tr> </table> <p>Your Platform Application will be created</p>	Platform Application Actions ▾	Application ARN arn:aws:sns:us-east-1:406334898880:app/GCM/Bootcamp2015-CognitoPush	Platform GCM
Platform Application Actions ▾				
Application ARN arn:aws:sns:us-east-1:406334898880:app/GCM/Bootcamp2015-CognitoPush				
Platform GCM				

Task 1-2: Create another Platform Application for the Mobile App—for Android

Overview	<p>We have created one Platform Application which will be used by Cognito Sync, but we also need to be able to send messages to our mobile application from our Lambda function. To do this, we will use a second SNS platform application.</p> <p>In this section we will create another SNS Platform Application that will be used for sending Push notifications to our Android application</p> <p><i>If you are not using Android, you should skip this section and instead locate the section for iOS SNS mobile push, in this same lab guide</i></p>
-----------------	--

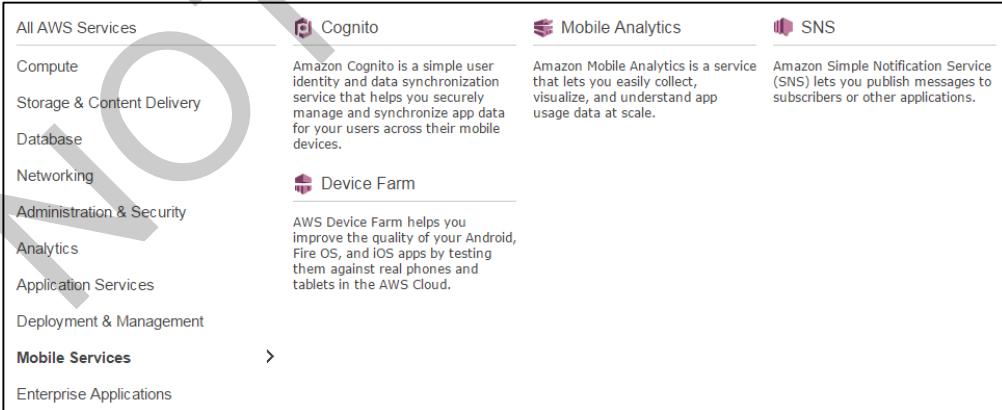
Step	Instruction
1.2.1	<p>Browse back to the SNS Home Page by clicking the SNS Home link in the navigation area.</p> 

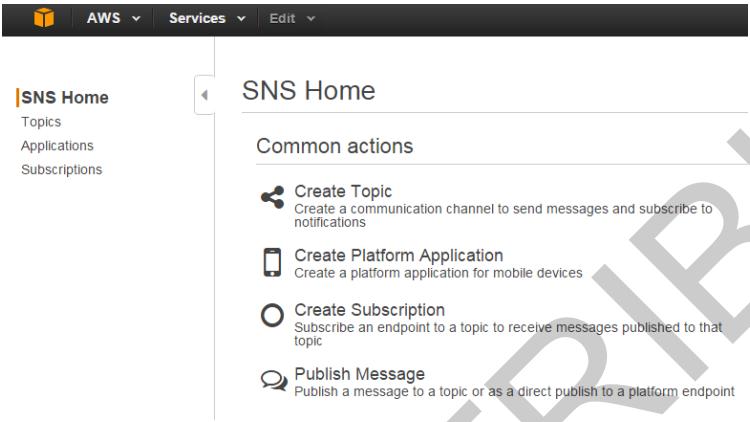
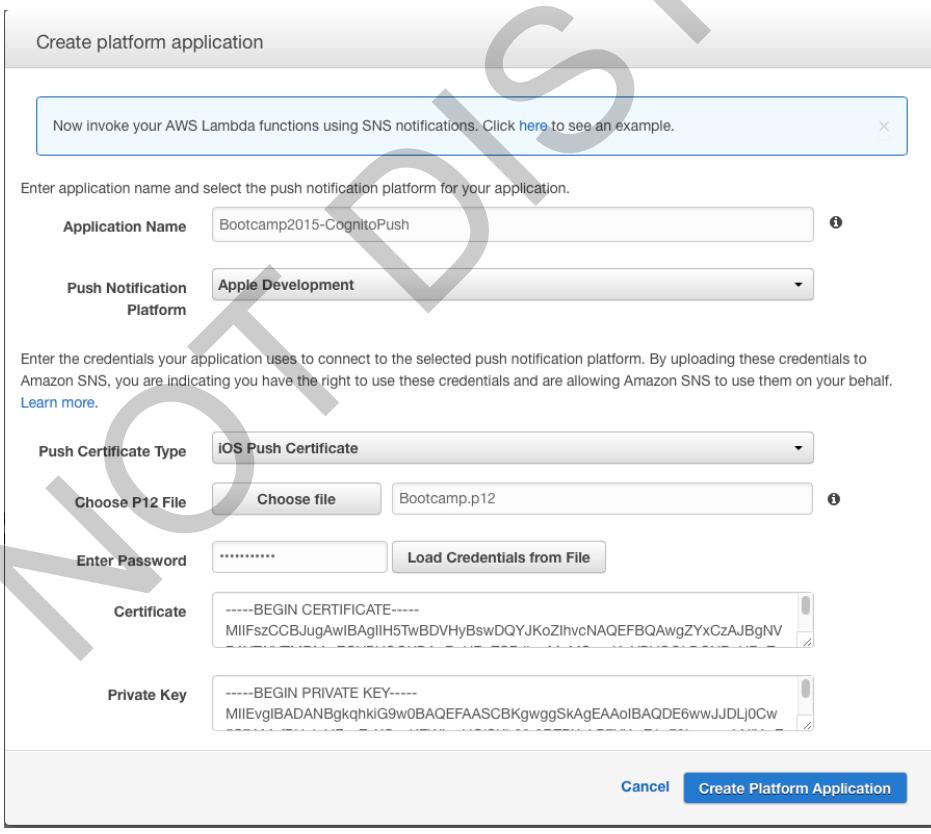
1.2.2	<p>In the AWS console for SNS, click the Create platform application link in the main panel.</p> 
1.2.3	<p>Name the platform application Bootcamp2015</p>  <p>Select the Push Notification platform. For Android, you must use Google Cloud Messaging (GCM)</p>

1.2.4	<p>Enter the API key from your Google console. If you don't have the API key handy, you can get the API key from your Google Developer Console by browsing to https://console.developers.google.com/project</p> <p><i>Note: As part of the prerequisites, we requested that you create a Google Project prior to attending the Bootcamp. If you have not done this, you will need to pause here, and follow the steps in the prerequisites information</i></p> <p>Paste your API Key, which will look something like this:</p> <pre>AlzaSyDdF-3XXXsXXXXsXXXXjNGzKx1GHb3xxUBJU</pre> <p>Click Create Platform Application</p>						
1.2.5	<p>Application Details: Bootcamp2015 (GCM)</p>  <table border="1" data-bbox="355 882 1318 1051"> <thead> <tr> <th colspan="2">Platform Application Actions</th> </tr> </thead> <tbody> <tr> <td>Application ARN</td> <td>arn:aws:sns:us-east-1:593576873474:app/GCM/Bootcamp2015</td> </tr> <tr> <td>Platform</td> <td>GCM</td> </tr> </tbody> </table> <p>Your Platform Application will be created.</p>	Platform Application Actions		Application ARN	arn:aws:sns:us-east-1:593576873474:app/GCM/Bootcamp2015	Platform	GCM
Platform Application Actions							
Application ARN	arn:aws:sns:us-east-1:593576873474:app/GCM/Bootcamp2015						
Platform	GCM						

Task 1-3: Create New Platform Applications for Push Synchronization—for iOS

Overview	<p>In this section we will create a new SNS Platform Application that will be used for Cognito Push Sync with iOS.</p> <p><i>If you are not using iOS, you should skip this section and instead locate the section for Android SNS mobile push, in this same lab guide</i></p>
-----------------	--

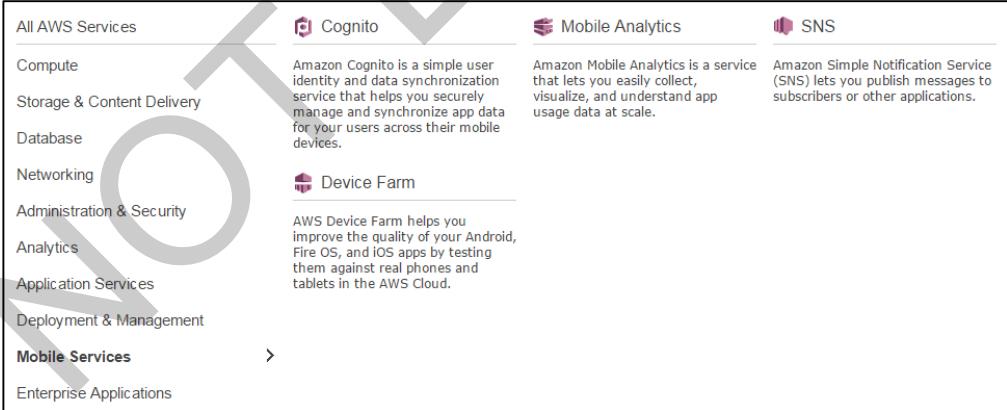
Step	Instruction
1.3.1	<p>Before starting with this section you should have completed the iOS application setup as a prerequisite to the Bootcamp and exported the private key as an *.p12 file. If you have not completed this process yet you must do that now. Locate the *.p12 file and keep the password you assigned during the export handy (the file is called ‘Bootcamp.p12’ and the password is ‘bootcamp123’).</p>
1.3.2	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services→Mobile Services→SNS.</p> 

1.3.3	<p>In the AWS console for SNS, click the Create platform application link in the main panel.</p> 
1.3.4	<p>Name the platform application Bootcamp2015-CognitoPush</p>  <p>Select the Push Notification platform. For iOS we will select Apple Development as the platform.</p>

1.3.5	Leave the Push Certificate Type as 'iOS Push Certificate' and click 'Choose File'. Browse to the Bootcamp.p12 file (or, whatever you named the private key when it was exported) and select it. Enter the password in the box below (such as 'bootcamp123') and press Load Credentials from File . If the Certificate and Private Key information populate, it means you have set up SNS to use your application with APNS. Press Create Platform Application .
-------	---

Task 1-4: Create another Platform Application for the Mobile App—for iOS

Overview	<p>We have created one Platform Application which will be used by Cognito Sync, but we also need to be able to send messages to our mobile application from our Lambda function. To do this, we will use a second SNS platform application.</p> <p>In this section we will create another SNS Platform Application that will be used for sending Push notifications to the iOS mobile application</p> <p><i>If you are not using iOS, you should skip this section and instead locate the section for Android SNS mobile push, in this same lab guide</i></p>
-----------------	---

Step	Instruction
1.4.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services→Mobile Services→SNS button:</p> 

- 1.4.2 In the AWS console for SNS, click the **Create platform application** link in the main panel.

The screenshot shows the AWS SNS Home page. At the top, there's a navigation bar with the AWS logo, Services dropdown, and Edit button. On the left, there's a sidebar with links for Topics, Applications, and Subscriptions. The main area is titled "SNS Home" and contains a "Common actions" section with four items: "Create Topic" (Create a communication channel to send messages and subscribe to notifications), "Create Platform Application" (Create a platform application for mobile devices), "Create Subscription" (Subscribe an endpoint to a topic to receive messages published to that topic), and "Publish Message" (Publish a message to a topic or as a direct publish to a platform endpoint). The "Create Platform Application" link is highlighted.

- 1.4.3 Name the platform application **Bootcamp2015**

The screenshot shows the "Create platform application" dialog box. It has a header "Create platform application" and a note: "Now invoke your AWS Lambda functions using SNS notifications. Click [here](#) to see an example." Below that, it says "Enter application name and select the push notification platform for your application." There are fields for "Application Name" (set to "Bootcamp2015") and "Push Notification Platform" (set to "Apple Development"). A note below says: "Enter the credentials your application uses to connect to the selected push notification platform. By uploading these credentials to Amazon SNS, you are indicating you have the right to use these credentials and are allowing Amazon SNS to use them on your behalf." It includes a "Learn more." link. The "Push Certificate Type" is set to "iOS Push Certificate". There are fields for "Choose P12 File" (with "Bootcamp.p12" selected) and "Enter Password". Below that are fields for "Certificate" (containing certificate text) and "Private Key" (containing private key text). At the bottom are "Cancel" and "Create Platform Application" buttons.

Select the Push Notification platform. For iOS we will select **Apple Development** as the platform.

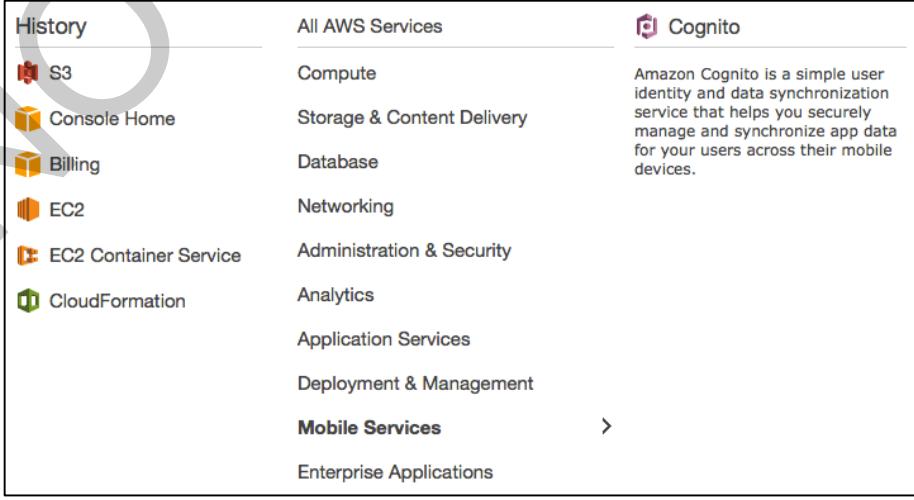
1.4.4	Leave the Push Certificate Type as 'iOS Push Certificate' and click 'Choose File'. Browse to the Bootcamp.p12 file (or whatever you named the private key when it was exported) and select it. Enter the password in the box below (such as 'bootcamp123') and press Load Credentials from File. If the Certificate and Private Key information populate, it means you have set up SNS to use your application with APNS. Press Create Platform Application .
-------	--

Task 2: Enable Push Synchronization in Cognito

Overview	In this task, we will enable Push Synchronization in our Cognito Identity Pool so that when a change is made to a dataset, the application can be automatically notified
-----------------	--

Task 2-1: Enabling Push Synchronization in Cognito

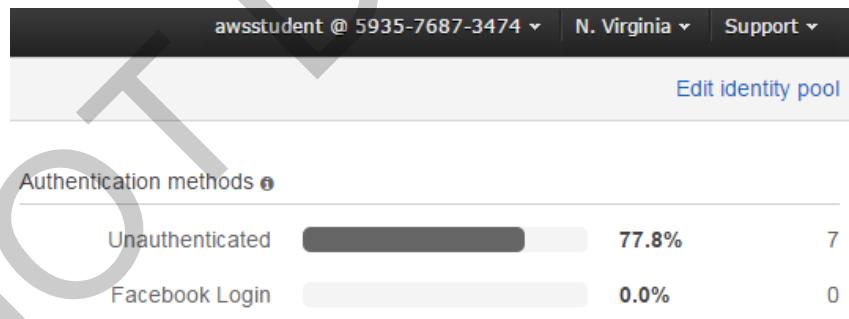
Overview	In the AWS console for Cognito you will modify the Cognito Identity Pool to use SNS Mobile Push for synchronizing datasets with the mobile application.
-----------------	---

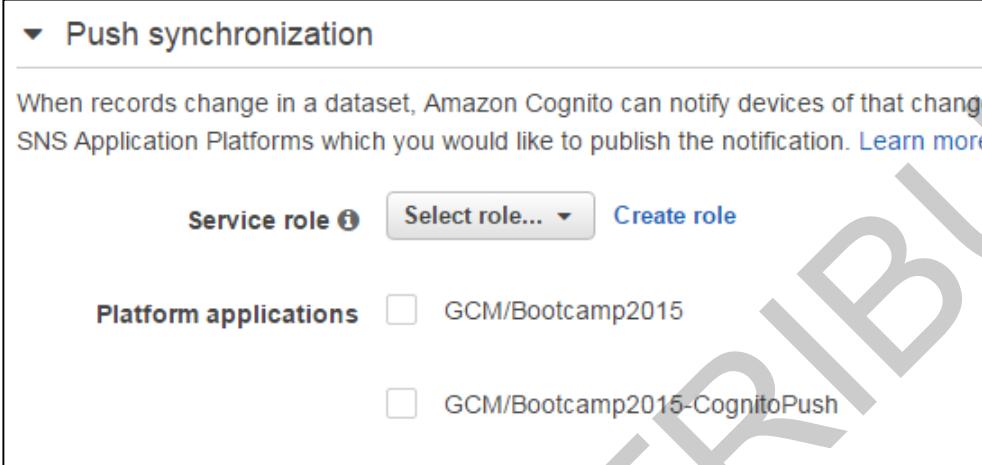
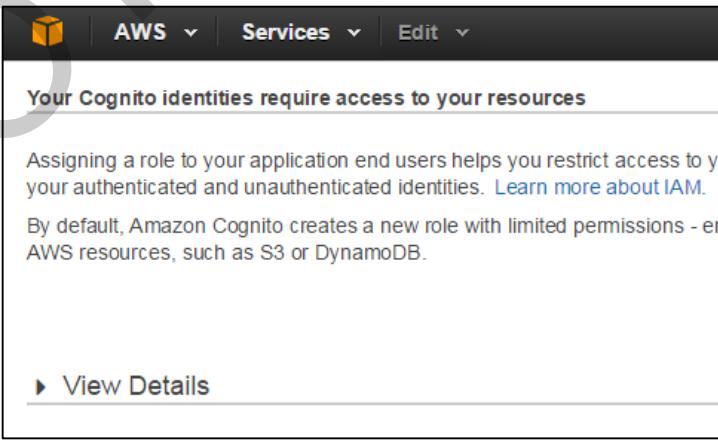
Step	Instruction
2.1.1	<p>Login to the AWS Management Console from your Qwiklabs Environment and choose Services→Mobile Services→Cognito.</p>  <p>The screenshot shows the AWS Management Console navigation bar. On the left, there's a sidebar with links to History, S3, Console Home, Billing, EC2, EC2 Container Service, CloudFormation, All AWS Services, Compute, Storage & Content Delivery, Database, Networking, Administration & Security, Analytics, Application Services, Deployment & Management, Mobile Services (which is highlighted in blue), and Enterprise Applications. To the right of the sidebar is a large area containing the Cognito service details, which are described in the text below.</p> <p>Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices.</p>

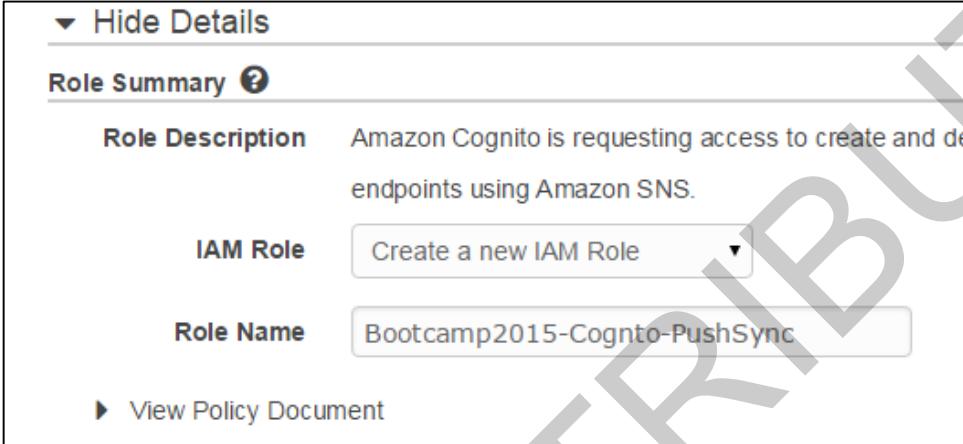
- 2.1.2 Click the **Bootcamp2015** heading link to open the Bootcamp2015 Cognito Identity Pool



- 2.1.3 In the top-right corner, click the **Edit Identity Pool** link.



2.1.4	<p>Scroll down to the Push Synchronization section.</p> 
2.1.5	<p>In the Service role section, click Create role.</p>  <p>This will assist us to create a new IAM role that Cognito can use to publish SNS messages</p>
2.1.6	<p>The page will refresh, and show a View Details section.</p>  <p>Click the 'twister' (arrow icon) to reveal the IAM panel</p>

2.1.7	<p>In the Role Name entry field, enter Bootcamp2015-Cognito-PushSyn as the name.</p> 
2.1.8	<p>In the bottom-right corner, click the Allow button.</p> 
2.1.9	<p>The page will refresh and redirect you back to the Cognito Identity Pool. Scroll down to the Push Synchronization section again, and click the twister to open the panel</p>

2.1.10	<p>Select the role we just created from the Service role list</p> <div data-bbox="388 367 1334 762"> <p>Push synchronization</p> <p>When records change in a dataset, Amazon Cognito can notify devices of that change using a SNS Application Platforms which you would like to publish the notification. Learn more about publishing notifications</p> <p>Service role Bootcamp2015-Cognito-PushSync Create role</p> <p>Platform applications</p> <table border="1"> <tr> <td><input type="checkbox"/> GCM/Bootcamp2015</td> </tr> <tr> <td><input checked="" type="checkbox"/> GCM/Bootcamp2015-CognitoPush</td> </tr> </table> </div> <p>Check the checkbox against the Platform Application you created earlier for push sync:</p> <p>Bootcamp2015-CognitoPush</p> <p>This instructs Cognito to send a notification to your app via the SNS Platform Application whenever a change is made to the dataset in the Cognito Sync Store.</p>	<input type="checkbox"/> GCM/Bootcamp2015	<input checked="" type="checkbox"/> GCM/Bootcamp2015-CognitoPush
<input type="checkbox"/> GCM/Bootcamp2015			
<input checked="" type="checkbox"/> GCM/Bootcamp2015-CognitoPush			
2.1.11	<p>Scroll down to the bottom right of the page, and click the Save changes button</p> <div data-bbox="659 1241 1062 1332"> <p>Cancel Save Changes</p> </div>		

Task 3: Building the Mobile Application from Source

Overview

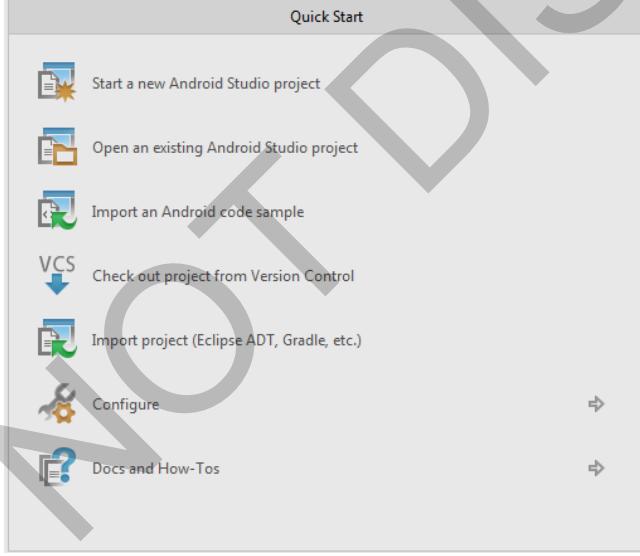
In this task, use XCode for iOS or Android Studio for Android builds of the Companion Mobile Application for our Connected Appliances System.

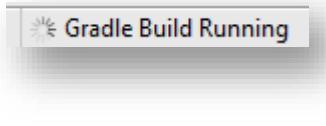
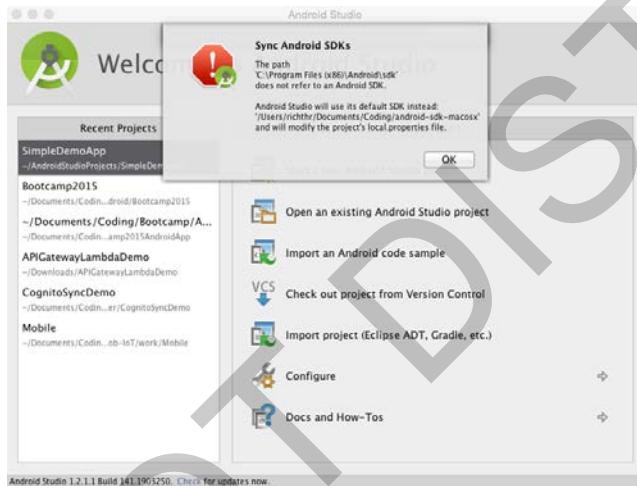
You should choose the tasks relevant to the mobile platform you are targeting.

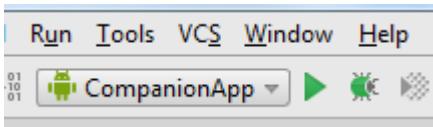
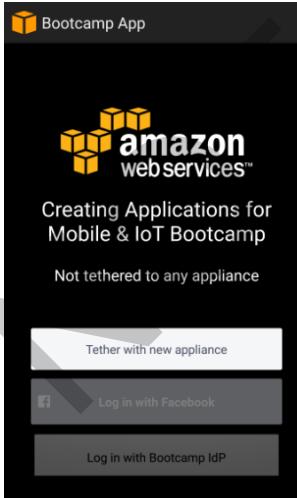
Task 3-1: Run the App for Android

Overview

In this task we will build the mobile application from the source code in Android Studio and debug/test on your mobile device.

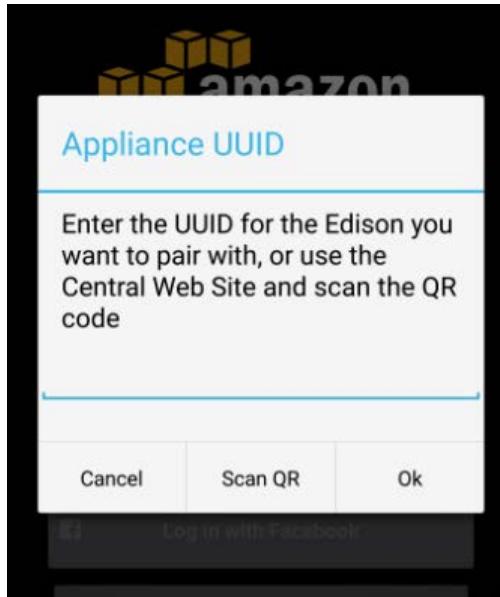
Step	Instruction
3.1.1	<p>First, we need to download the source code zip file and unpack it on the local computer. In your browser, hit the given URL that will download the Android source code for our Companion Mobile App:</p> <p>http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/AndroidSource.zip</p> <p><i>Note: On a Mac you may need to run “curl –O http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/AndroidSource.zip”</i></p> <p>Unpack the zip file contents to a folder location on your computer that you will use to build using Android Studio. We will refer to this folder as the source code folder, later in this guide</p>
3.1.2	<p>Start Android Studio. From the Quick Start panel, choose Open an existing Android Studio project.</p> 

3.1.3	<p>Browse to the location of the source code folder you unpacked the contents of the zip file into and click OK.</p> <p>Android Studio will import the project and start up the IDE. The project will then automatically download any missing dependencies and perform a build. This may take a few minutes</p>  <p>If you see an SDK Path error like below press OK to finish launching and accept the default SDK Path.</p> 
3.1.4	<p>When the build is complete, you can plug in your mobile device into a spare USB socket with a suitable cable. Note that you need to have turned on Developer Mode on your Android device. For details on that, we recommend you search Google for <i>how to turn on android developer mode</i> because each mobile device could be different.</p> <p><i>You also have the option of using the Android Studio device emulator instead of a physical device. In this case, when prompted for a device to launch your application on, select the appropriate emulator.</i></p>

3.1.5	<p>When you are ready to start the build and test on your device, make sure CompanionApp is selected in the configuration list, and click the debug button (the bug icon) to start a build and debug session</p>  <p>After a minute or two of build, you will see some log information in the Android Studio console, such as:</p> <pre>Waiting for device. Target device: samsung-sm_g925i-03157df324ad3227 Uploading file local path: C:\temp\DesktopBootcampThursday\Bootcamp2015\CompanionApp\build\outputs\apk\CompanionApp-debug.apk remote path: /data/local/tmp/au.com.emersent.bootcamp2015.apps.companionapp Installing au.com.emersent.bootcamp2015.apps.companionapp DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/au.com.emersent.bootcamp2015.apps.companionapp"</pre>
3.1.6	<p>The app will be deployed to your device, and it will start running. On your device, you will see the Login Activity, which is the launcher activity for our app.</p>  <p>The login buttons are greyed out and disabled because you cannot log in to the app until you are tethered with an Edison appliance.</p>

3.1.7

Touch the **Tether with new appliance** button to start the tethering process



The Appliance UUID dialog will appear. You can either enter the Appliance UUID of your Edison device manually by typing it into the Android keyboard, or, if you have the **ZXing Barcode Scanner** installed on your device, you can use the Scan QR option.

If you want to install the ZXing barcode scanner, visit the Play Store. For information about the ZXing barcode scanner, see

<https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=en>

- 3.1.8 Browse to the Central website (<http://bit.ly/mobile-iot-bootcamp>) and locate your device by the name you gave it in the bootcamp-config.js file in Lab guide 1. You can use the Filter feature to enter in the name of your device to help locate your Edison in the list. The item will look like this:

	Adam's Edison	Kettle	984fee033fbc	192.168.200.20	badf00d5dff48d2b060d0f5a71096e0
---	---------------	--------	--------------	----------------	---------------------------------

Manual UUID Entry:

If you don't have the ZXing barcode scanner installed, you can manually enter the Appliance UUID characters using the keyboard. Touch in the text entry field and the keyboard will be displayed.

QR Scan Option:

If you have the ZXing Barcode Scanner installed, touch the Scan QR option in the dialog. On the Central website page, click the row in the list that identifies your device, and a details panel will appear, containing a QR Code for your device. You can scan using your camera



Whether you enter the UUID manually or scan it, the outcome is the same

3.1.9



Now the application has tethered with your Edison, it is ready for you to log in. But before you do, we will start up the appliance manager on the Edison(s).

Because we are working in pairs, you and your partner should run the appliance manager code on your Edisons so that the devices discover each other and publish the detections to our DynamoDB table, because this table is what will populate our 'Nearby Appliances' list in your app.

On the primary and secondary Edisons, run the appliance manager node application

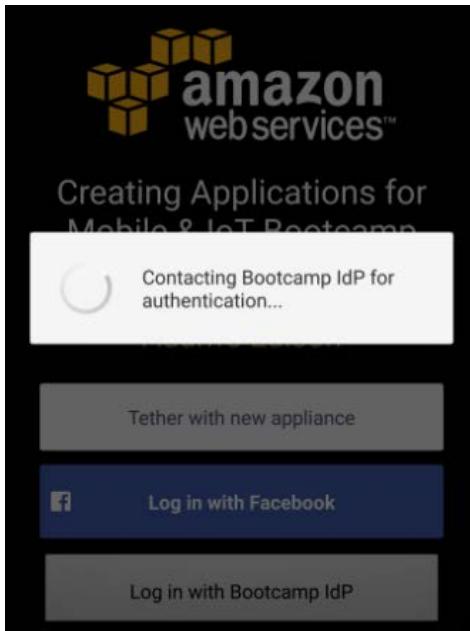
```
cd /home/root/labs  
node appliance-manager.js
```

This will start the appliance manager. If both the Primary and Secondary have the same BLE channel (UUID, as set in the bootcamp-config.js on each Edison) then they will detect each other and start sending detection data to Kinesis, as you have seen throughout the Bootcamp.

```
root@mini:~/labs# node appliance-manager.js  
  
*****  
**  
** Bootcamp2015 Appliance Manager  
**  
*****  
  
onNobleStateChange -> poweredOn  
Starting to Advertise this Edison...  
Initialising Cognito...  
Cognito IdentityId => us-east-1:8a4e45b8-3b62-4dc8-bd43-71f24f8620ae  
Starting to listen for other Edisons...  
BLE MAC ADDRESS = 984fee033fbc  
New Edison discovered: 001122335566 @ -64  
192.168.200.22:1(001122335566) :: 0 dBm >> 0 metres away  
192.168.200.22:1(001122335566) :: -61.4753 dBm >> 1.1100 metres away  
192.168.200.22:1(001122335566) :: -60.8746 dBm >> 1.0730 metres away  
192.168.200.22:1(001122335566) :: -61.1079 dBm >> 1.0871 metres away  
192.168.200.22:1(001122335566) :: -61.3688 dBm >> 1.1032 metres away
```

3.1.10	<p>Now the Edisons are detecting each other and sending the detection data to DynamoDB via Kinesis and Lambda we can log in to our app</p> <p>Touch the Log in with Bootcamp IdP button</p>
3.1.11	<p>The app displays a login dialog box.</p> <p>The dialog is pre-populated with username and password; leave these as they are, and touch the OK button.</p>

3.1.12



The app will show a modal progress dialog while it connects with our authentication Lambda functions, using the API Gateway we set up previously

3.1.13



When authentication has completed, the app will show the Main Activity, which shows the Primary Edison (the Edison we have tethered this mobile device to) and a list of the nearby appliances below.

Here you can see our appliance is a kettle, and we have a toaster nearby, just less than a metre away.

Touch the 'Refresh' button a few times, to see the distance calculation and RSSI update in 'real time'.

3.1.14



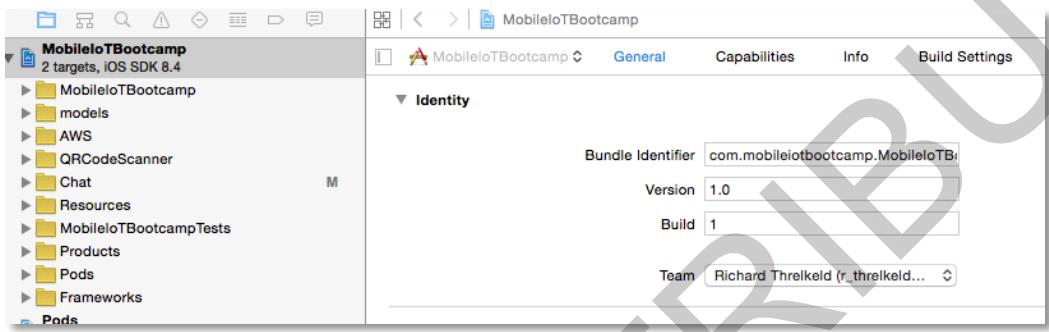
A few moments after the Main Activity is shown, the device will have had enough time to complete the registration with the application's SNS topic (which we created earlier and called **Bootcamp2015-To-MobileApp**) and it tests the connection by publishing a welcome message to itself. The app receives the push notification and displays the message as a pop-up.

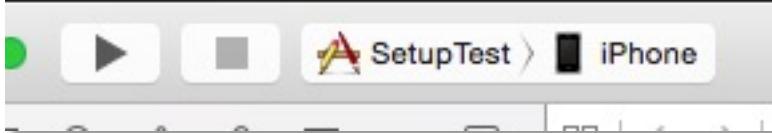
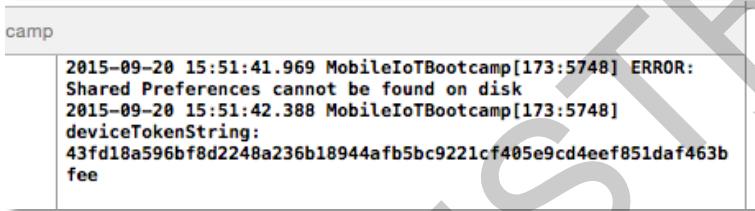
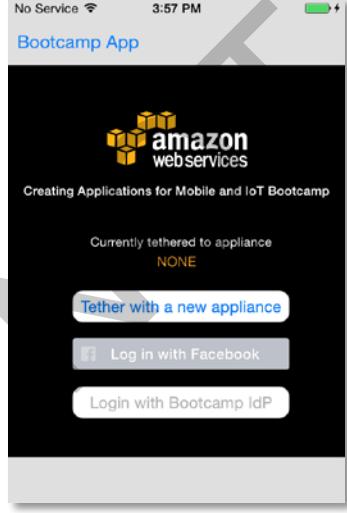
You can dismiss this message by touching anywhere outside the dialog on the screen.

Task 3-1: Run the App for iOS

Overview	In this task we will build the mobile application from the source code in XCode and debug/test on your mobile device.
-----------------	---

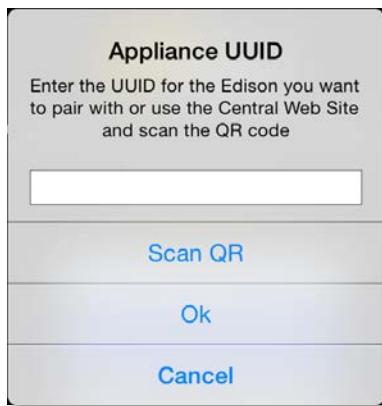
Step	Instruction
3.1.1	<p>First, we need to download the source code zip file and unpack it on the local computer. In your browser, hit the given URL that will download the iOS source code for our Companion Mobile App:</p> <p>http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/iOSSource.zip</p> <p><i>Note: On a Mac you may need to run “curl -O http://bootcamp2015-mobile-iot.s3.amazonaws.com/bundle/iOSSource.zip”</i></p> <p>Unpack the zip file contents to a folder location on your computer that you will use to build using XCode. We will refer to this folder as the source code folder, later in this guide.</p>

3.1.2	<p>In the source code folder open the MobileIoTBootcamp.xcworkspace file (NOT the MobileIoTBootcamp.xcodeproj file!) and click the top-level “MobileIoTBootcamp” section in the upper left corner. You should be in the project properties “General” section.</p>  <p>Locate the “Bundle Identifier” and ensure that this matches the Bundle Identifier you created in the Apple Developer Member Center from the Bootcamp prerequisites documentation. You should just need to insert the UNIQUEID after “com.mobileiotbootcamp.” in XCode.</p>
3.1.3	<p>Quit XCode (XCode→Quit XCode) and in a Terminal window navigate to the source code folder and run:</p> <pre>\$ pod install</pre> <p>If everything runs fine you should see dependencies analyzed for AWS and Facebook SDKs.</p>

3.1.4	<p>In the source code folder open the MobileIoTBootcamp.xcworkspace file again and plug in your powered off iPhone to your Mac. Turn on the iPhone and if you can select it as a deployment target as seen below then it has been successfully detected by XCode.</p>  <p>A message about the device token should be printed out to the console in XCode.</p> 
3.1.5	<p>The app will be deployed to your device and will be running. On your device you will see the Tether and Login screen.</p>  <p>The login buttons are greyed out until a successful tether with an Edison appliance takes place.</p>

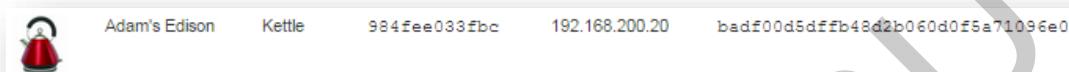
3.1.6

Touch the **Tether with a new appliance** button to start the tethering process.



The Appliance UUID dialog will appear. You can either enter the Appliance UUID from the Central website or scan it by selecting 'Scan QR'. You will need to grant the App permissions to use the camera if you select the Scan QR option.

- 3.1.7 Browse to the Central website (<http://bit.ly/mobile-iot-bootcamp>) and locate your device by the name you gave it in the bootcamp-config.js file in Lab guide 1. You can use the Filter feature to enter in the name of your device to help locate your Edison in the list. The item will look like this:

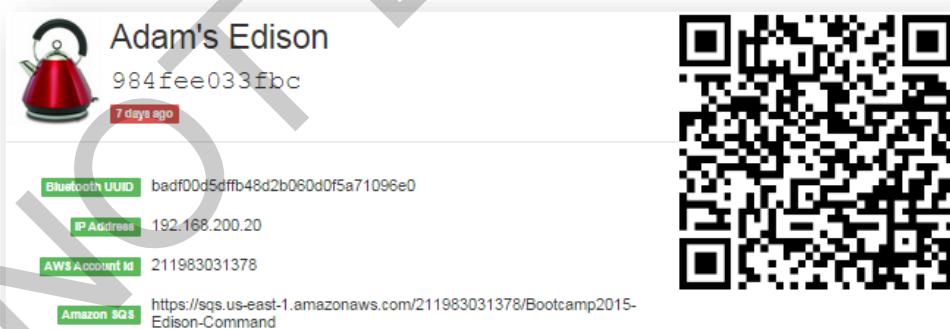


Manual UUID Entry:

If you don't use the Scan QR option, you can manually enter the Appliance UUID characters using the keyboard. Touch in the text entry field and the keyboard will be displayed.

QR Scan Option:

If you don't use manual entry, touch the Scan QR option in the dialog. On the Central website page, click the row in the list that identifies your device, and a details panel will appear, containing a QR Code for your device. You can scan using your camera.

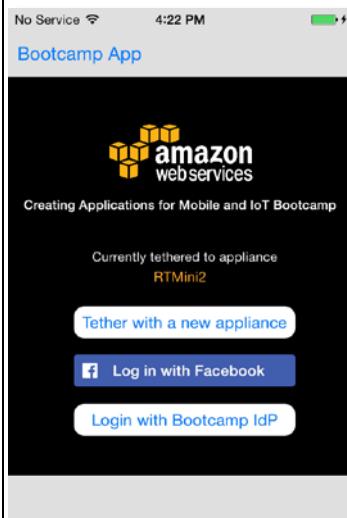


Whether you enter the UUID manually or scan it, the outcome is the same.

3.1.8

iOS requires whenever a tether happens (in this section and when re-tethering in the future) you must close and reopen the application. After tethering the application will close automatically. After this take place re-open the Bootcamp application either from your iOS homepage or with XCode.

3.1.9



Now the application has tethered with your Edison, it is ready for you to log in. But before you do, we will start up the appliance manager on the Edison(s).

Because we are working in pairs, you and your partner should run the appliance manager code on your Edisons so that the devices discover each other and publish the detections to our DynamoDB table, because this table is what will populate our 'Nearby Appliances' list in your app.

On the primary and secondary Edisons, run the appliance manager node application with

```
cd /home/root/labs  
node appliance-manager.js
```

This will start the appliance manager. If both the Primary and Secondary have the same BLE channel (UUID, as set in the bootcamp-config.js on each Edison) then they will detect each other and start sending detection data to Kinesis, as you have seen throughout the Bootcamp.

```
root@mini:~/labs# node appliance-manager.js  
  
*****  
**  
** Bootcamp2015 Appliance Manager  
**  
*****  
  
onNobleStateChange -> poweredOn  
Starting to Advertise this Edison...  
Initialising Cognito...  
Cognito IdentityId => us-east-1:8a4e45b8-3b62-4dc8-bd43-71f24f8620ae  
Starting to listen for other Edisons...  
BLE MAC ADDRESS = 984fee033fbc  
New Edison discovered: 001122335566 @ -64  
192.168.200.22:1(001122335566) :: 0 dBm >> 0 metres away  
192.168.200.22:1(001122335566) :: -61.4753 dBm >> 1.1100 metres away  
192.168.200.22:1(001122335566) :: -60.8746 dBm >> 1.0730 metres away  
192.168.200.22:1(001122335566) :: -61.1079 dBm >> 1.0871 metres away  
192.168.200.22:1(001122335566) :: -61.3688 dBm >> 1.1032 metres away
```

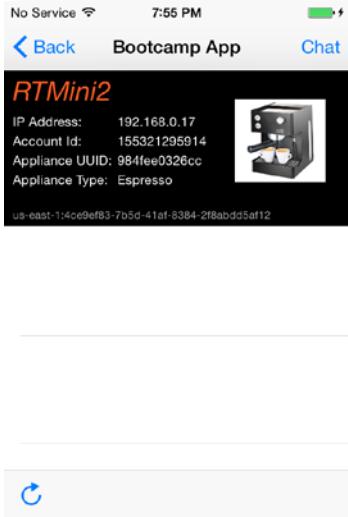
3.1.10	<p>Now the Edisons are detecting each other and sending the detection data to DynamoDB via Kinesis and Lambda; we can log in to our app.</p> <p>Touch the Log in with Bootcamp IdP button</p>
3.1.11	<p>The app displays a login dialog.</p> <p>The dialog is pre-populated with a username and password; leave these as they are and press the Login button.</p>

- 3.1.12 If all the Apple Developer Member Center tasks for creating an App have been completed with a certificate uploaded to SNS, and the unique Bundle ID has been matched in XCode then a moment after authenticating you should see a welcome message, which the application published to itself.



If you did not see this welcome message validate the Bundle ID matches that which was entered into the Apple Developer Member Center and that the SNS Platform Application for APNS has been configured in the **Bootcamp2015-To-MobileApp** SNS Topic correctly.

- 3.1.13 The application will show tethered appliance information at the top and an empty device list. It should also show the device Cognito ID in light grey at the bottom of the black window at the top.

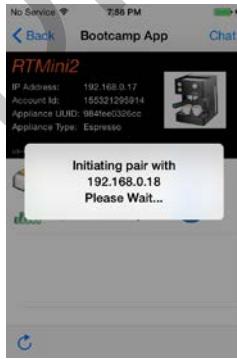


Touch the refresh arrow in the lower left of the screen to see a list of the nearby devices as populated in DynamoDB from the Lambda functions you created earlier. If you touch this several times you will see the RSSI update in 'real-time'.



Task 3-2: Test the Chat Feature

Overview	In this task, we take a look at the chat feature of our application. All the required source code and configuration has been done for you, you just need to test if it works. This will confirm that all the SNS topics, Lambda functions, etc. have been configured to the correct specification.
-----------------	--

Step	Instruction
3.2.1	<p>From the Main Activity, touch the pairing button (the 'link' icon) next to the appliance you want to pair with. Android devices will show a modal wait dialog while the pairing process is completed.</p>  <p>iOS devices have a similar message using the Apple alerting framework.</p> 

3.2.2	<p>On the Primary Edison mobile app (the app that you pressed the Pair icon on) you will see the following output.</p> <pre>*** COMMANDPAIRREQUEST received from SQS -> Pair with 192.168.200.18 3fbc) onPairMessageRequestReceived --> 984fee033fbc Attempting MQTT connection to 192.168.200.18 (984fee033fbc) Connected to 192.168.200.18 Subscribing to topic 'edison' Now subscribed to topic 'edison' on mqtt://192.168.200.18 Publishing COMMAND_ANNOUNCE to topic COMMAND_ANNOUNCE sent ok via MQTT Informing back end of current pair peers... ipAddress=192.168.200.18 - uuid=984fee033fbc SNS sending asynchronously to "arn:aws:sns:us-east-1:404965379528:Bootcamp2015-From-Edison" onMQTTClientLocalMessageReceived -> COMMAND_ANNOUNCE_ACK SNS sending asynchronously to "arn:aws:sns:us-east-1:404965379528:Bootcamp2015-From-Edison" 192.168.200.18:5(984fee033fbc) :: -53.3173 dBm >> 0.3632 metres away 192.168.200.18:5(984fee033fbc) :: -53.0394 dBm >> 0.3447 metres away SNS sent ok [onRemoteDeviceAnnounce] SNS sent ok</pre>
3.2.3	<p>And the secondary will show the following at the same time.</p> <pre>onMQTTClientLocalMessageReceived -> COMMAND_ANNOUNCE SNS sending asynchronously to "arn:aws:sns:us-east-1:532236681459:Bootcamp2015-From-Edison" onPairMessageRequestReceived --> 984fee0325d7 Attempting MQTT connection to 192.168.200.29 (984fee0325d7) Connected to 192.168.200.29 Subscribing to topic 'edison' Now subscribed to topic 'edison' on mqtt://192.168.200.29 Publishing COMMAND_ANNOUNCE to topic COMMAND_ANNOUNCE sent ok via MQTT Informing back end of current pair peers... ipAddress=192.168.200.29 - uuid=984fee0325d7 SNS sending asynchronously to "arn:aws:sns:us-east-1:532236681459:Bootcamp2015-From-Edison" 192.168.200.29:5(984fee0325d7) :: -53.0178 dBm >> 0.3433 metres away 192.168.200.29:5(984fee0325d7) :: -52.7933 dBm >> 0.3291 metres away SNS sent ok [onRemoteDeviceAnnounce] SNS sent ok</pre>

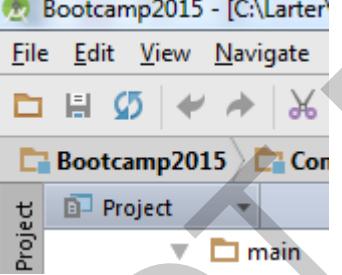
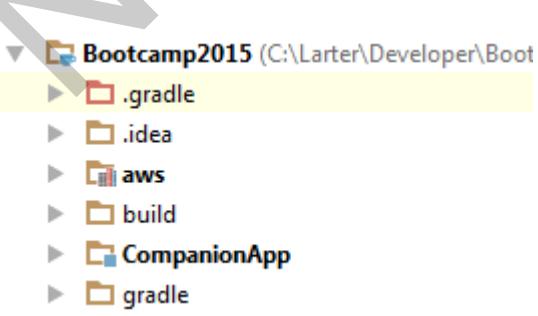
3.2.4	<p>This illustrates the dialog between the Edisons as they convey information.</p> <p>When the pairing is complete, you can run the Chat feature in both Primary and Secondary companion mobile apps. Press the Chat button in the upper right-hand corner to start the Chat feature.</p> <p>On one app, enter “Hello”</p> <p>Moments later, as the message is transferred from Mobile App→SQS→Edison→MQTT→Edison→SNS→Lambda→SNS→Mobile App, you will see the message appear on the other (Secondary) mobile app</p> <p>You can then reply on the Secondary, and it will appear on the Primary when transferred</p> <p>While the message and the ACKs are in transit, you will see activity between the Edisons that will be shown on the SSH session consoles</p>
-------	---

Task 4: Implementing the Appliance Control Panels

Overview	<p>In this task, we will edit the mobile application code to implement some missing functionality—the appliance control panels</p> <p>The mobile app that has been provided to you does not have a complete implementation of the control panel feature. The part that is missing is the synchronization with the Cognito Sync Manager. You will implement this missing code in this task</p> <p>Use XCode for iOS or Android Studio for Android builds of the Companion Mobile Application for our Connected Appliances System.</p> <p>You should choose the tasks relevant to the mobile platform you are targeting.</p>
-----------------	--

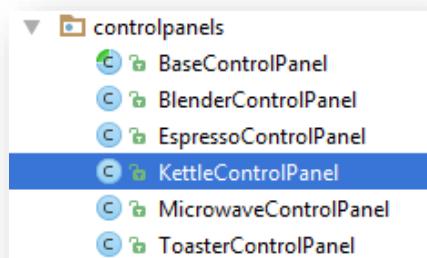
Task 4-1: Implement the Control Panels – Android Studio

Overview	<p>In this task, we will implement the ability of the mobile user to show a control panel for each appliance they can see in their appliances list. The control panel shows various appliance-specific settings and persists the settings to the Cognito Sync Store in a Dataset.</p> <p>Implement the code, build the mobile application in Android Studio and debug/test on your mobile device.</p>
-----------------	---

Step	Instruction
4.1.1	<p>In Android Studio, ensure that the Project view is selected in the solution explorer panel on the left.</p>  <p>Locate the CompanionApp folder in the solution explorer panel, and expand it to reveal <code>CompanionApp src main java controlpanels</code></p> 

4.1.2	<p>In the <i>controlpanels</i> folder you will see several Java classes – one for each type of control panel our app supports, and a <i>BaseControlPanel</i>, from which all the other control panels derive</p> <pre data-bbox="344 460 752 804"> ▶️ 📁 command └ 📁 controlpanels 🐾 BaseControlPanel 🐾 BlenderControlPanel 🐾 EspressoControlPanel 🐾 KettleControlPanel 🐾 MicrowaveControlPanel 🐾 ToasterControlPanel └ 📁 messaging </pre> <p>Open the <i>BaseControlPanel</i> by double clicking on it</p>
4.1.3	<p>Search for the function <i>synchronizeSharedData()</i> using CTRL+F</p> <pre data-bbox="323 973 1307 1453"> protected void synchronizeSharedData(boolean bUpdateUIOnSuccess) { // // TODO: Implement an override of this method in the derived Control panel // classes, to synchronize the state of the gadgets with the // Cognito Dataset // new Handler(Looper.getMainLooper()) .post(() -> { Toast.makeText(BaseControlPanel.this, "Sync has not been implemented for this Control Panel!", Toast.LENGTH_LONG).show(); }); } </pre> <p>Notice that this implementation pops up a Toaster if it is called. The source code you have been supplied does not have an implementation of overrides of this function in the derived class therefore no actual synchronization will occur. Your task is to implement the overrides where applicable</p>

4.1.4 If you have time, you can implement overrides for all the control panel types. However, we suggest you start with the appliance type that your partner has defined as their appliance type. The reason for this is, only your partner's appliance will show in your nearby appliance list, so if you want to be able to use the control panel, you will need to implement the control panel for their particular appliance.



Later, if time permits, you can pair with other students in the class, and so may need to implement other control panels based on the appliance types they have chosen

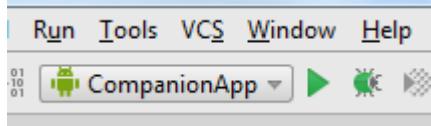
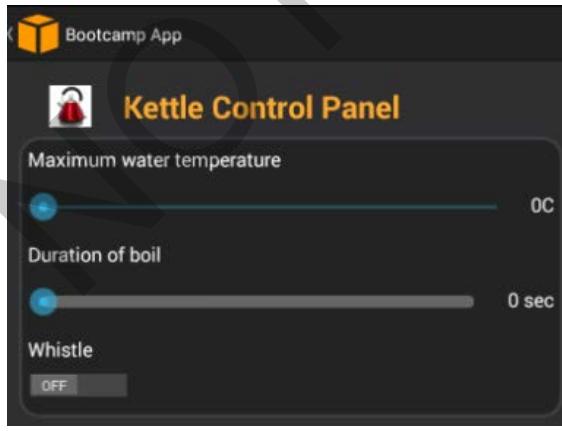
Let's assume you need to implement the **Kettle** control panel.

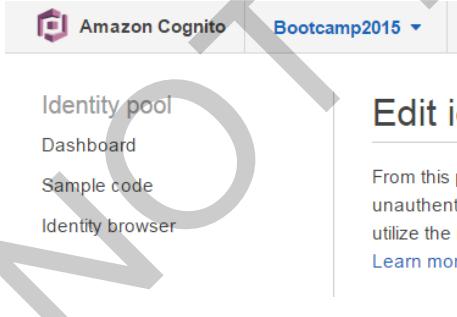
Double-click on the KettleControlPanel file in the solution explorer

4.1.5 The code you need to implement is shown here:

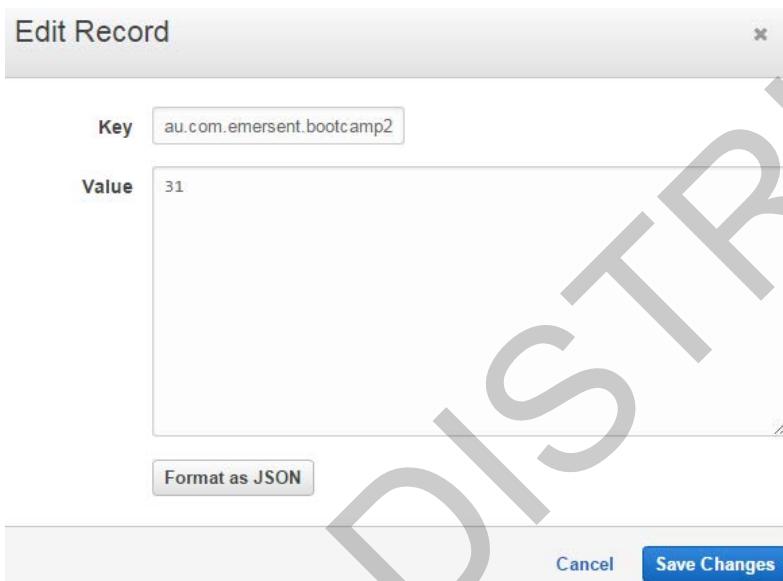
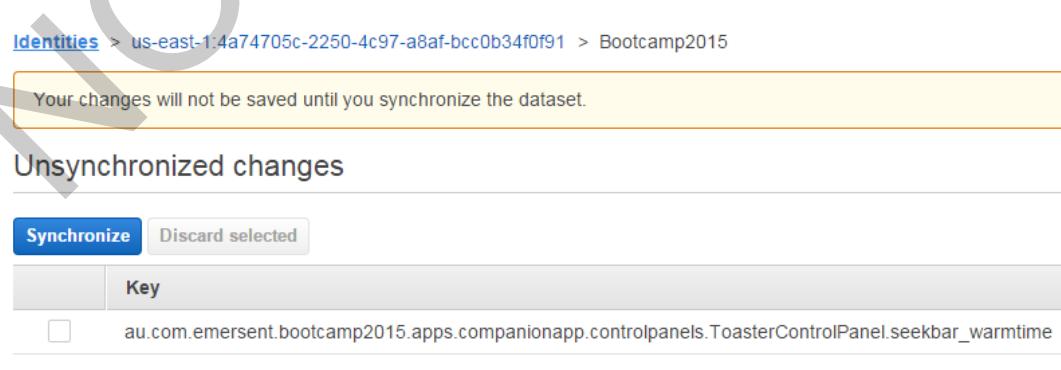
```
@Override  
protected void synchronizeSharedData(boolean bUpdateUIOnSuccess)  
{  
    Cognito.getDataset().put(  
        this.getClass().getName() + ".seekbar_temperature",  
        "" + CalculateTemperatureSliderValueFromProgress(this.seekbar_temperature.getProgress())  
    );  
  
    Cognito.getDataset().put(  
        this.getClass().getName() + ".seekbar_boiltime",  
        "" + CalculateBoilTimeSliderValueFromProgress(this.seekbar_boiltime.getProgress())  
    );  
  
    Cognito.getDataset().put(  
        this.getClass().getName() + ".switch_whistle",  
        "" + this.switch_whistle.isChecked()  
    );  
  
    synchroniseCognito(bUpdateUIOnSuccess);  
}
```

Note that it is a protected override of the void function `synchronizeSharedData()` taking a Boolean object in.

4.1.6	<p>To implement this function, locate the <code>Bootcamp2015-Android-ControlPanel-Kettle.txt</code> file in the Desktop bundle you downloaded in Lab 1.</p> <p>Paste the contents of this file into an appropriate location in the <code>KettleControlPanel</code> implementation file. For example, you could paste the function implementation just before the last curly brace in the file, if you like.</p>
4.1.7	<p>When you are ready to start the build and test on your device, make sure CompanionApp is selected in the configuration list, and click the debug button (the bug icon) to start a build and debug session</p>  <p>The app will build and run as before (unless you made a mistake during the pasting in of the function)</p> <p>Log in again (note that you don't need to re-tether) and this time, in the Main Activity, touch the Control Panel icon (cogs icon) for your partner's appliance.</p> <p>The control panel will appear. Here is an example of the Kettle Control Panel.</p> 

4.1.8	<p>If you adjust the values in the control panel, a Cognito Sync will occur, by executing the synchronizeSharedData() function you just implemented in this control panel. To confirm the data is being synchronized, we will inspect the Cognito Sync Store for this user identity.</p> <p>From the Control panel, press the Android device's Back button to reveal the Main Activity again.</p> <p>At the top of the Main Activity, the Cognito Identity for the currently logged-on user is shown:</p>  <p>Make a note of the Id; you will need it in the next steps</p>
4.1.9	Navigate to the Cognito page in your AWS Management Console and drill into the Cognito Identity Pool for our bootcamp – Bootcamp2015
4.1.10	<p>Choose Identity Browser from the links on your right</p> 
4.1.11	<p>In the Search by Identity ID field, enter the Cognito identity from the Main Activity in the mobile app. Click the Search button.</p> <p>Identities</p> <p>Search by Identity ID us-east-1:4a74705c-2250-4c97-a8af-bcc0b34f0f91 Search</p> <p>Tip: You can also search by your developer identifier.</p>

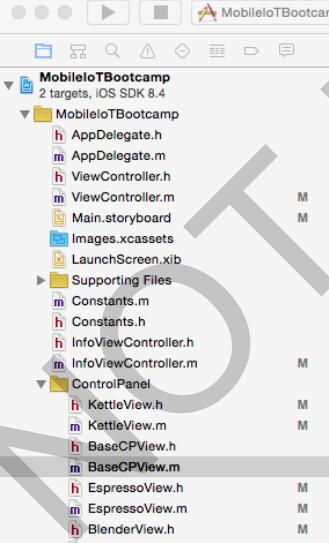
4.1.12	<p>When the Identity is found, click it to drill in. A list of the datasets will be shown below the general information.</p> <p>Click on the dataset name to drill into the contents of the dataset</p> <p>Identities > us-east-1:4a74705c-2250-4c97-a8af-bcc0b34f0f91</p> <p>Identity details</p> <p>Delete identity</p> <p>Date created (UTC) 2015-07-11T10:00:17Z</p> <p>Linked logins graph.facebook.com</p> <p>Datasets</p> <table border="1"> <thead> <tr> <th></th><th>Dataset name</th><th>Date created (UTC)</th><th>Last modified (UTC)</th></tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td><td>Bootcamp2015</td><td>2015-07-11T10:03:35Z</td><td>2015-07-11T10:03:37Z</td></tr> </tbody> </table>		Dataset name	Date created (UTC)	Last modified (UTC)	<input type="checkbox"/>	Bootcamp2015	2015-07-11T10:03:35Z	2015-07-11T10:03:37Z	
	Dataset name	Date created (UTC)	Last modified (UTC)							
<input type="checkbox"/>	Bootcamp2015	2015-07-11T10:03:35Z	2015-07-11T10:03:37Z							
4.1.13	<p>Notice that the values you set on the Android code implementation of synchronizeSharedData() have been pushed into the Cognito Sync Store.</p> <p>Identities > us-east-1:4a74705c-2250-4c97-a8af-bcc0b34f0f91 > Bootcamp2015</p> <p>Current dataset - Bootcamp2015</p> <table border="1"> <thead> <tr> <th></th> <th>Key</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>au.com.emersent.bootcamp2015.apps.companionapp.controlpanels.ToasterControlPanel.seekbar_toastdarkness</td> <td>54</td> </tr> <tr> <td><input type="checkbox"/></td> <td>au.com.emersent.bootcamp2015.apps.companionapp.controlpanels.ToasterControlPanel.seekbar_warmtime</td> <td>31</td> </tr> </tbody> </table>		Key	Value	<input type="checkbox"/>	au.com.emersent.bootcamp2015.apps.companionapp.controlpanels.ToasterControlPanel.seekbar_toastdarkness	54	<input type="checkbox"/>	au.com.emersent.bootcamp2015.apps.companionapp.controlpanels.ToasterControlPanel.seekbar_warmtime	31
	Key	Value								
<input type="checkbox"/>	au.com.emersent.bootcamp2015.apps.companionapp.controlpanels.ToasterControlPanel.seekbar_toastdarkness	54								
<input type="checkbox"/>	au.com.emersent.bootcamp2015.apps.companionapp.controlpanels.ToasterControlPanel.seekbar_warmtime	31								

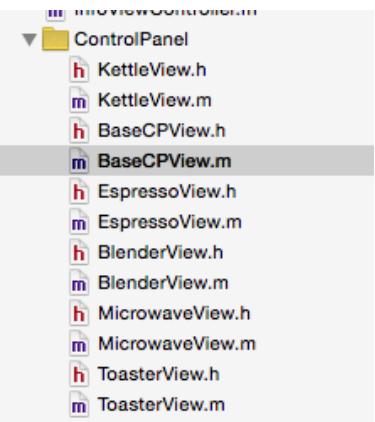
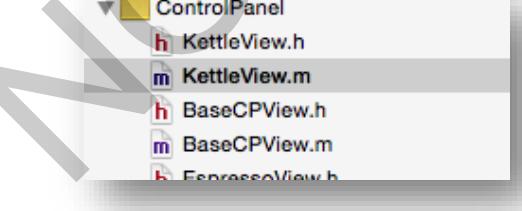
4.1.14	<p>Now, make a change to the dataset directly in the Sync Store. For example, if you were dealing with a Toaster control panel, you could click on seekbar_warmtime.</p> <p>Change the value to any value other than the current one—it is recommended that you choose a value between 10 and 100 as a general rule.</p>  <p>Click Save Changes</p>
4.1.15	<p>Note that the change won't be saved until you synchronize. Click the Synchronize button</p> 

4.1.16	<p>A moment later, you should see your Mobile app automatically synchronize with Cognito, as a result of you changing the dataset.</p> <p>Cognito Push Sync has sent a notification to the configured Platform Application, to which our mobile application has registered. The app therefore receives a push notification when the dataset changes. It handles this by refreshing the dataset and updating the mobile app's user interface.</p> <p>This is a good demonstration of how you can make ad-hoc, programmatic or manual changes to the dataset, and have that dataset updated across platforms and devices, using Cognito Push Sync!</p>
--------	--

Task 4-1: Implement the Control Panels – Xcode

Overview	<p>In this task, we will implement the ability of the mobile user to show a control panel for each appliance they can see in their appliances list. The control panel shows various appliance-specific settings and persists the settings to the Cognito Sync Store in a Dataset.</p> <p>Implement the code, build the mobile application in XCode and debug/test on your mobile device.</p>
-----------------	--

Step	Instruction
4.1.1	<p>In XCode, click on the Project Navigator in the upper left hand corner. Expand the MobileIoTBootcamp ControlPanel folder.</p> 

4.1.2	<p>In the <i>controlpanels</i> folder you will see several Objective-C classes – one for each type of control panel our app supports, and a <i>BaseCPView</i>, from which all the other control panels derive.</p>  <p>Your job will be to implement Cognito Sync in the derived implementation classes.</p>
4.1.3	<p>If you have time, you can implement sync for all the control panel types. However, we suggest you start with the appliance type that your partner has defined as their appliance type. The reason for this is, only your partner's appliance will show in your nearby appliance list, so if you want to be able to use the control panel, you will need to implement the control panel for their particular appliance.</p> <p>Later, if time permits, you can pair with other students in the class, and so may need to implement other control panels based on the appliance types they have chosen</p>  <p>Let's assume you need to implement the Kettle control panel.</p> <p>Click on the <i>KettleView.m</i> file in the solution explorer</p>

4.1.4	<p>The code you need to implement is shown here:</p> <pre><code>- (IBAction)updateButton:(id)sender { AWSCognito *syncClient = [AWSCognito defaultCognito]; NSUserDefaults *preferences = [NSUserDefaults standardUserDefaults]; //Sync Cognito Dataset AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:[preferences objectForKey:@"uniqueDatasetName"]]; NSLog(@"Synchronizing dataset with Cognito"); [dataset synchronize]; } @end</code></pre> <p>Note that this method is an IBAction that is already linked to the Storyboard so that dataset synchronization only happens when a user presses the 'Update' button.</p>
4.1.5	<p>To implement this function, locate the <code>Bootcamp2015-iOS-ControlPanel-Kettle.txt</code> file in the Desktop bundle you downloaded in Lab 1.</p> <p>Locate the <code>-(IBAction)updateButton:</code> method towards the end of the file and paste the contents of the method implementation from this file into the method implementation in <code>KettleView.m</code>. <u>Do not copy and paste over the method or you may break the Storyboard connection.</u></p>

4.1.6

When you are ready to start the build and test on your device, make sure **iPhone** is selected and press the arrow to the left.

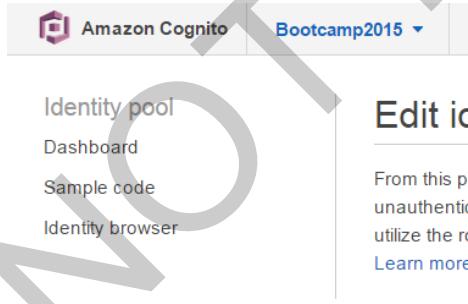


The app will build and run as before (unless you made a mistake during the pasting in of the function)

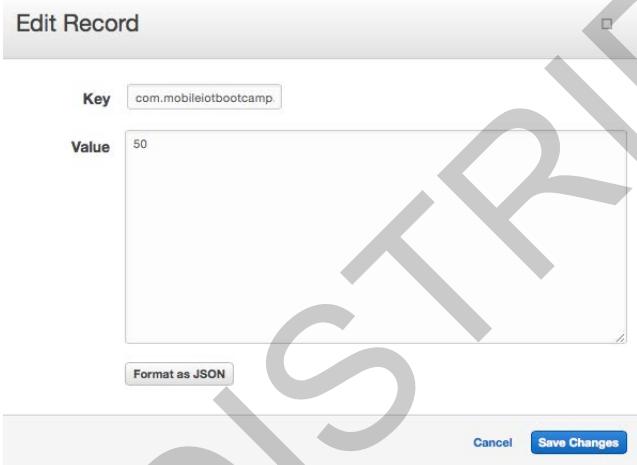
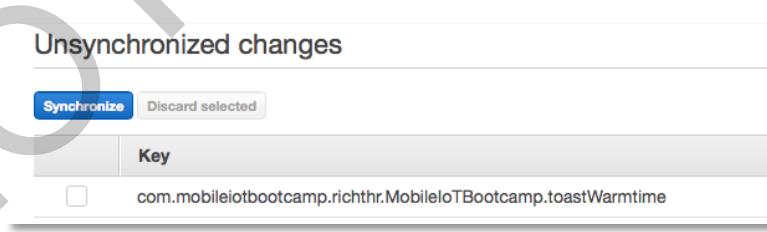
Log in again (note that you don't need to re-tether) and this time, in the Main Activity, touch the Control Panel icon (cogs icon) for your partner's appliance.

The control panel will appear. Here is an example of the Toaster Control Panel.



4.1.7	<p>If you adjust the values in the control panel and press 'Update', a Cognito Sync will occur, by executing the –(IBAaction)updateButton: method you just implemented in this control panel. To confirm the data is being synchronized, we will inspect the Cognito Sync Store for this user identity.</p> <p>From the Control panel, press the iOS device's Back button to reveal the Main View again.</p> <p>At the top of the Main View, the Cognito Identity for the currently logged-on user is shown:</p>  <p>Make a note of the Id; you will need it in the next steps</p>
4.1.8	<p>Navigate to the Cognito page in your AWS Management Console and drill into the Cognito Identity Pool for our bootcamp – Bootcamp2015</p>
4.1.9	<p>Choose Identity Browser from the links on your right</p> 
4.1.10	<p>In the Search by Identity ID field, enter the Cognito identity from the Main Activity in the mobile app. Click the Search button.</p> <p>Identities</p> <p>Search by Identity ID <input type="text" value="us-east-1:4a74705c-2250-4c97-a8af-bcc0b34f0f91"/> Search</p> <p>Tip: You can also search by your developer identifier.</p>

4.1.11	<p>When the Identity is found, click it to drill in. A list of the datasets will be shown below the general information.</p> <p>Click on the dataset name to drill into the contents of the dataset</p> <p>Identities > us-east-1:4a74705c-2250-4c97-a8af-bcc0b34f0f91</p> <p>Identity details</p> <p>Delete identity</p> <p>Date created (UTC) 2015-07-11T10:00:17Z</p> <p>Linked logins graph.facebook.com</p> <p>Datasets</p> <table border="1"> <thead> <tr> <th></th><th>Dataset name</th><th>Date created (UTC)</th><th>Last modified (UTC)</th></tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td><td>Bootcamp2015</td><td>2015-07-11T10:03:35Z</td><td>2015-07-11T10:03:37Z</td></tr> </tbody> </table>		Dataset name	Date created (UTC)	Last modified (UTC)	<input type="checkbox"/>	Bootcamp2015	2015-07-11T10:03:35Z	2015-07-11T10:03:37Z	
	Dataset name	Date created (UTC)	Last modified (UTC)							
<input type="checkbox"/>	Bootcamp2015	2015-07-11T10:03:35Z	2015-07-11T10:03:37Z							
4.1.12	<p>Notice that the values set on the Objective-C code have been pushed into the Cognito Sync Store. The values utilize the bundle ID you configured earlier in XCode.</p> <table border="1"> <thead> <tr> <th></th> <th>Key</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td><td>com.mobileiotbootcamp.richthr.MobileIoTBootcamp.toastWarmtime</td><td>50</td></tr> <tr> <td><input type="checkbox"/></td><td>com.mobileiotbootcamp.richthr.MobileIoTBootcamp.waterTemp</td><td>43</td></tr> </tbody> </table>		Key	Value	<input type="checkbox"/>	com.mobileiotbootcamp.richthr.MobileIoTBootcamp.toastWarmtime	50	<input type="checkbox"/>	com.mobileiotbootcamp.richthr.MobileIoTBootcamp.waterTemp	43
	Key	Value								
<input type="checkbox"/>	com.mobileiotbootcamp.richthr.MobileIoTBootcamp.toastWarmtime	50								
<input type="checkbox"/>	com.mobileiotbootcamp.richthr.MobileIoTBootcamp.waterTemp	43								

4.1.13	<p>Now, make a change to the dataset directly in the Sync Store. For example, if you were dealing with a Toaster control panel, you could click on toastWarmtime.</p> <p>Change the value to any value other than the current one—it is recommended that you choose a value between 10 and 100 as a general rule.</p>  <p>Click Save Changes</p>
4.1.14	<p>Note that the change won't be saved until you synchronize. Click the Synchronize button.</p> 

4.1.15	<p>A moment later, you should see your Mobile app automatically synchronize with Cognito, as a result of you changing the dataset.</p> <p>Cognito Push Sync has sent a notification to the configured Platform Application, to which our mobile application has registered. The app therefore receives a push notification when the dataset changes. It handles this by refreshing the dataset and updating the mobile app's user interface.</p> <p>This is a good demonstration of how you can make ad-hoc, programmatic or manual changes to the dataset, and have that dataset updated across platforms and devices, using Cognito Push Sync!</p>
--------	--

Lab 9: Bonus Lab – Implementing Mobile Analytics

Overview

This is a bonus lab. If you have completed the previous labs during the Bootcamp in the allocated time, you may want to attempt this Lab guide, which shows you how to use Amazon Mobile Analytics.

One of the biggest challenges for mobile developers is retaining users. Studies estimate that nearly 60% of users never come back to an app after downloading it and firing it up for the first time.

Mobile Analytics helps you track user retention once the app is installed on their device. Developers can use the information gathered by Amazon Mobile Analytics to optimize their app experience or track the effectiveness of a marketing campaign.

With Amazon Mobile Analytics you can get behavioral insights into app specific actions that your users take. For example, if you are a game developer you'd be interested in tracking user abort rates in each level of the game. If you build a music streaming app, you'd be interested in tracking the number of songs played per active user. If you build a news reader app, you'd be interested in tracking the popularity of news articles.

The Mobile Analytics Dashboard provides a view of how often these custom events occur, and you can add further context with Attributes and Metrics.

Amazon Mobile Analytics works cross-platform for iOS, Android, or Fire OS apps using the SDKs, or using the API to submit events. User engagement can be tracked with KPIs such as Active Users or Sessions per active user. Monetization trends are captured in terms of in-app revenue and LTV metrics.

In our system, we create and use Custom events to send various

	<p>metrics from attributes of the Control Panels to Amazon Mobile Analytics. For example, Toaster – keep-warm time, Kettle – boil time, and we also record In-app purchases by simulating a charge each time a Control Panel is raised by the user, and sending the event to Amazon Mobile Analytics so we can plot the average revenue per user in the dashboard.</p> <p>In this module, we will implement the mobile analytics into our companion app. We will record relatively simple data, however this will give you a good understanding of how you can extend the mobile analytics facility to capture rich analytics data across your application.</p> <p>We will also record the number of seconds each of the appliance control panels are visible to each user. This will tell us how often these panels are used and for how long, and allow us to use this data to make changes to our app to make it better.</p> <p>This Lab is not intended to teach you how to be an Android/iOS programmer, but instead, assumes you have experience with these platforms and focuses on how to use the AWS SDKs to integrate your applications and to take advantage of the AWS Mobile Optimized Services and Connectors</p> <p><i>Before commencing this Lab guide, you must have previously installed Android Studio or XCode depending on the platform you are targeting. See the Bootcamp prerequisites documentation</i></p> <p><i>It is also assumed that you have completed Lab Guide 8 prior to commencing this Lab Guide</i></p>
--	--

Objectives	<p>After completing this lab, you will be able to:</p> <ul style="list-style-type: none"> • Set up a new application for Mobile Analytics • Integrate mobile analytics in the companion mobile app • Access the analytics data in the console
-------------------	--

Pre-requisites	This lab requires: <ul style="list-style-type: none">Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).<ul style="list-style-type: none">Note The <i>qwikLABS</i> lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.For Microsoft Windows users: Administrator access to the computer.An Internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).
-----------------------	--

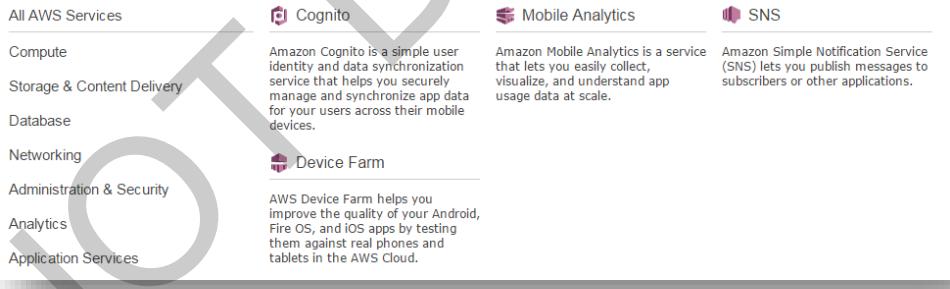
Duration	30 minutes
-----------------	------------

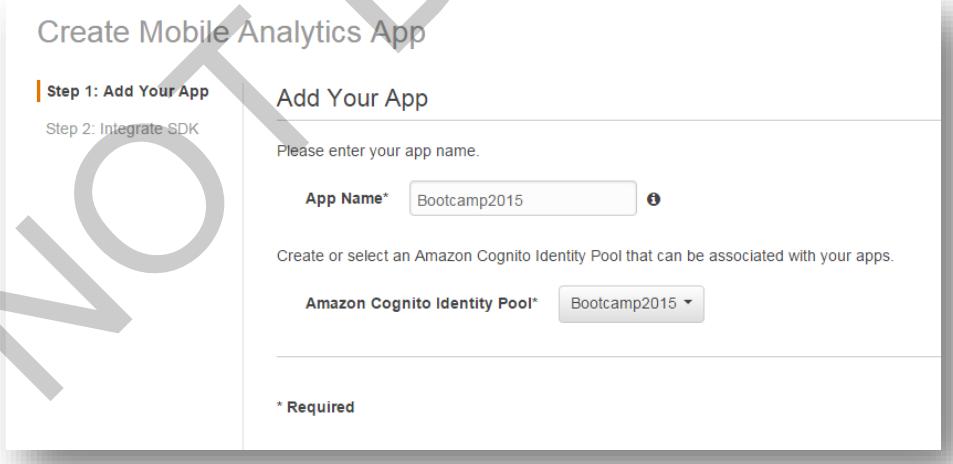
Task 1: Creating a Mobile Analytics Application

Overview

We will first create a Mobile Analytics Application that will be associated with your app.

Task 1-1: Create a Mobile Analytics Application

Step	Instruction
1.1.1	<p>Navigate to the Mobile Analytics dashboard under Mobile Services - Mobile Analytics.</p> 

1.1.2	<p>You will see the Get Started screen</p>  <p>Click Get Started</p>
1.1.3	<p>On the Create Mobile Analytics App page, name your app Bootcamp2015</p> <p>In the Amazon Cognito Identity Pool drop-down list, select our Bootcamp2015 for the identity pool, and press Create App.</p> 

- 1.1.4 Your analytics app is now created. You can take a look at the integration steps required to use mobile analytics in your application.

The screenshot shows the AWS Mobile Analytics integration steps for the application 'Bootcamp2015'. At the top, there's a navigation bar with 'AWS', 'Services', and 'Edit' dropdowns. Below it, the 'Amazon Mobile Analytics' icon and the app name 'Bootcamp2015' are displayed. A large heading 'Integration Steps for Bootcamp2015' is centered. A sub-instruction says 'Select or create an Amazon Cognito Identity Pool to associate with your app.' with a dropdown showing 'Bootcamp2015' and a blue button 'Create an Amazon Cognito Identity Pool'. Below this are tabs for 'iOS', 'Android', 'Unity', 'JavaScript', and 'Xamarin', with 'iOS' currently selected. The first step, '1. Include the SDK', has two options: 'CocoaPods' (selected) and 'Manual'. It instructs to update the Podfile with the command: `pod 'AWSMobileAnalytics', '~> 2.2.2'`. The second step, '2. Initialize Mobile Analytics', asks to add imports to AppDelegate with the command: `#import <AWSMobileAnalytics/AWSMobileAnalytics.h>`.

- 1.1.5 In the details of the integration, the **App Id** that you will need to use in your app, is shown in the code samples. For example, in the iOS sample code, the mobile analytics app Id is shown:

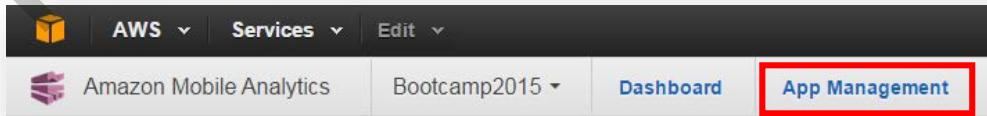
```
In the application:didFinishLaunchingWithOptions: method in the ApplicationDelegate for your app, use:
```

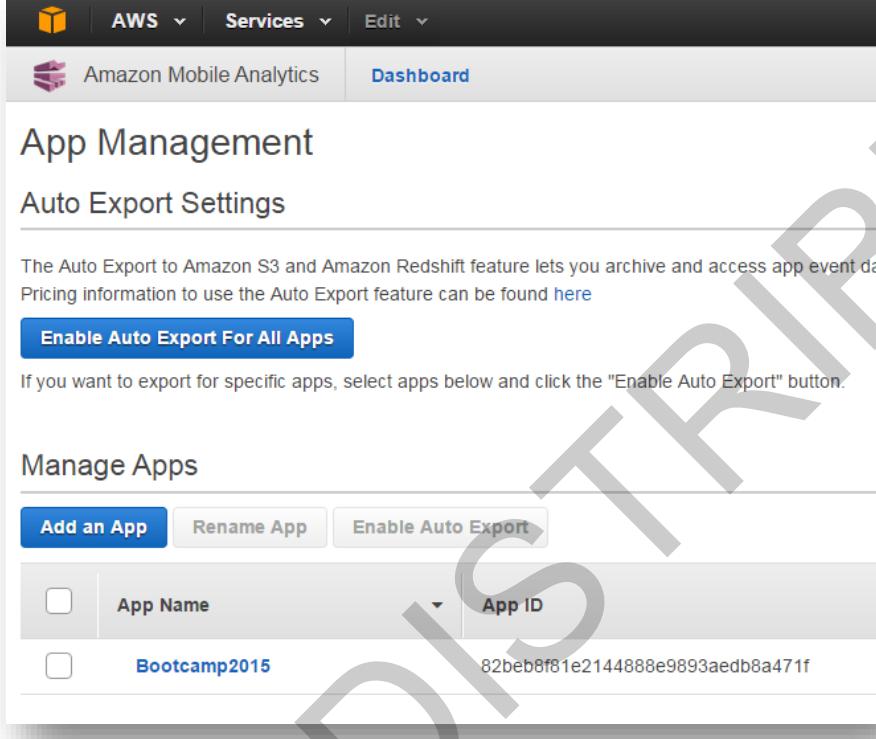
```
AWSMobileAnalytics *analytics = [AWSMobileAnalytics
    mobileAnalyticsForAppId: @"82beb8f81e2144888e9893aedb8a471f" //Amazon Mobile Analytics App ID
    identityPoolId: @"/us-east-1:1c3c8a68-d362-457f-ae43-3fae4e91d114"]; //Amazon Identity Pool ID
```

Similarly for Android:

```
try {
    analytics = MobileAnalyticsManager.getInstance(
        this.getApplicationContext(),
        "82beb8f81e2144888e9893aedb8a471f", //Amazon Mobile Analytics App ID
        "us-east-1:1c3c8a68-d362-457f-ae43-3fae4e91d114", //Amazon Identity Pool ID
    );
} catch(InitializationException ex) {
    Log.e(this.getClass().getName(), "Failed to initialize Amazon Mobile Analytics");
}
```

- 1.1.6 You can get access to the App Id at any time, but selecting App Management from the top menu of the Mobile Analytics console:



1.1.7	<p>All the apps you have created, and their App Ids, are available in the list:</p>  <table border="1" data-bbox="349 946 1225 1136"> <thead> <tr> <th>Add an App</th><th>Rename App</th><th>Enable Auto Export</th></tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td><td>App Name</td><td>App ID</td></tr> <tr> <td><input type="checkbox"/></td><td>Bootcamp2015</td><td>82beb8f81e2144888e9893aedb8a471f</td></tr> </tbody> </table>	Add an App	Rename App	Enable Auto Export	<input type="checkbox"/>	App Name	App ID	<input type="checkbox"/>	Bootcamp2015	82beb8f81e2144888e9893aedb8a471f
Add an App	Rename App	Enable Auto Export								
<input type="checkbox"/>	App Name	App ID								
<input type="checkbox"/>	Bootcamp2015	82beb8f81e2144888e9893aedb8a471f								
1.1.8	<p>We now need to add the App ID of the Mobile Analytics app you just created, into your bootcamp-config.js file, so that this information is published to the Central website, and subsequently made available to the mobile app during tethering.</p> <p>To do this, copy the app Id from the app management console (see 1.1.7) into your clipboard. Be sure to include the entire Id. It will look like this:</p> <pre>82beb8f81e2144888e9893aedb8a471f</pre> <p>But of course your Id will be different.</p>									

1.1.9	<p>With the app Id in the clipboard, go over to your Edison device terminal, enter the following in an SSH session:</p> <pre>vi /etc/bootcamp2015-config.js</pre> <p>root@mini3:~# vi /etc/bootcamp2015-config.js</p> <p>Navigate to the MOBILE_ANALYTICS_APP_ID definition, and add the App Id value:</p> <pre>/////////////////////////////// // Configure these values // /////////////////////////////// var DEVICE_NAME = "Mini Edison"; var AWS_ACCOUNT_ID = "0000000000"; var COGNITO_IDENTITY_POOL_ID = "us-east-1:XXXXXXXXXXXX"; var MOBILE_ANALYTICS_APP_ID = "0000000000"; var API_GATEWAY_ENDPOINT_URL = "https://REPLACE_THIS_WITH_API_GATEWAY_URL"; var APPLIANCE_TYPE = 5; var BLE_UUID = "badf00d5dfffb48d2b060d0f5a71096e0"; var GOOGLE_PROJECT_NUMBER = "0000000000";</pre> <p>To do this, use the cursor keys to position the cursor at the first 0 of the MOBILE_ANALYTICS_APP_ID value. Press the 'x' key as many times as required to delete all the zeroes, leaving just "" as the value</p> <p>Now, press the <i>i</i> key to go into Insert mode.</p> <p>Right-click your mouse to paste in the contents of the clipboard into the current cursor position. Your definition for MOBILE_ANALYTICS_APP_ID should now look like this:</p> <pre>var MOBILE_ANALYTICS_APP_ID = "82beb8f81e2144888e9893aedb8a471f";</pre> <p>But of course your Id will be different.</p>
-------	---

1.1.10	<p>Leave insert mode by pressing the ESC key</p> <p>Save the file by issuing the <code>:x</code> command, which will exit the file and return you to the main console.</p> <p>Now you need to instruct the Edison to re-synchronize with the Central website and store its updated configuration. On the Edison, enter</p> <pre>systemctl restart bootcamp-register.service</pre> <p>In a few moments, your Edison will register the changes with Central Web</p>
1.1.11	<p>You will need to re-tether your mobile app when you've published the app Id to the Central website. Check Lab Guide 8 for information about how to do that if you are unsure. If you are using iOS you will need to re-open the application after a successful tether takes place.</p>

Task 2: Implementing Mobile Analytics in Your Mobile Application

Overview

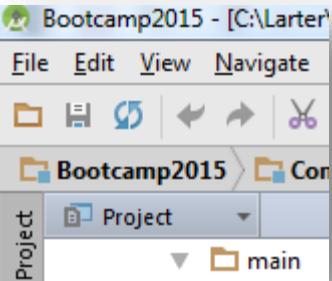
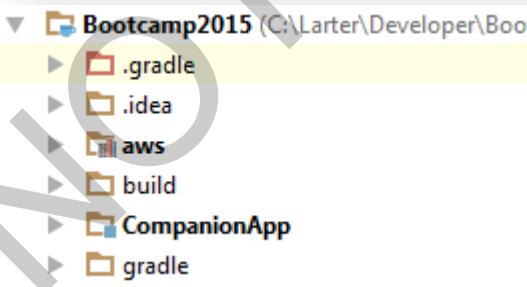
In this task, we will edit the mobile application code to implement some missing functionality—the mobile analytics integration.

The mobile app that has been provided to you has a partial implementation of the mobile analytics feature. The part that is missing is recording the actual events with the Mobile Analytics service. Each Control Panel has its own implementation for this feature, and you will implement the missing code in this task.

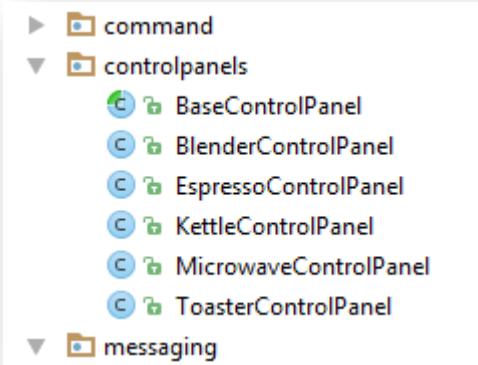
Use XCode for iOS or Android Studio for Android builds of the Companion Mobile Application for our Connected Appliances System.

You should choose the tasks relevant to the mobile platform you are targeting.

Task 2-1: Implement Mobile Analytics – Android Studio

Step	Instruction
2.1.1	<p>In Android Studio, ensure the Project view is selected in the solution explorer panel on the left:</p>  <p>Locate the CompanionApp folder in the solution explorer panel, and expand it to reveal <i>CompanionApp src main java controlpanels</i></p> 

- 2.1.2 In the *controlpanels* folder you will see several Java classes; one for each type of control panel that our app supports, and a *BaseControlPanel*, from which all the other control panels derive



Open the *BaseControlPanel* by double clicking on it

2.1.3

Search for the function RecordAnalytics() using CTRL+F

```
protected void RecordAnalytics()
{
    double countOfSecondsPanelWasVisible = (new Date().getTime() - datePanelStarted.getTime()) / 1000;

    AnalyticsEvent syncEvent = Cognito.getAnalytics().getEventClient().createEvent(getPanelNameForAnalytics())
        .withAttribute("identity", Cognito.getIdentityId())
        .withMetric("TimeVisible", countOfSecondsPanelWasVisible);

    Cognito.getAnalytics().getEventClient().recordEvent(syncEvent);
}
```

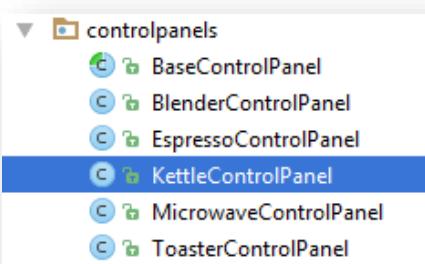
The base class' implementation records the length of time the control panel was open for. The base class' implementation is called each time the Control Panel is closed. You can confirm this by looking at the onStop() method in the BaseControlPanel:

```
@Override
public void onStop()
{
    super.onStop();

    /**
     * Record the stats for this panel
     */
    RecordAnalytics();
}
```

When onStop() is called by the Android framework, the derived Control Panel class' RecordAnalytics() method is called polymorphically.

Your task is to implement overrides of this method in the derived Control Panel classes, where applicable.

2.1.4	<p>If you have time, you can implement overrides for all the control panel types. However, we suggest you start with the appliance type that your partner has defined as their appliance type. The reason for this is, only your partner's appliance will show in your nearby appliance list, so if you want to be able to record the analytics for a control panel, you will need to implement the analytics functionality in the control panel for their particular appliance type.</p> <p>Later, if time permits, you can pair with other students in the class, and so may need to implement other control panels based on the appliance types they have chosen</p>  <p>Let's assume you need to implement the Kettle control panel.</p> <p>Double-click on the KettleControlPanel file in the solution explorer.</p>
2.1.5	<p>The code you need to implement is shown here:</p> <pre> /** * Records analytics specific to our control panel */ @Override protected void RecordAnalytics() { super.RecordAnalytics(); // // We record the temperature and duration so we can look at some averages in the console // AnalyticsEvent syncEvent = Cognito.getAnalytics().getEventClient().createEvent(getPanelNameForAnalytics()) .withMetric("MaxTemp", (double)CalculateTemperatureSliderValueFromProgress(this.seekbar_temperature.getProgress())); Cognito.getAnalytics().getEventClient().recordEvent(syncEvent); syncEvent = Cognito.getAnalytics().getEventClient().createEvent(getPanelNameForAnalytics()) .withMetric("BoilTime", (double)CalculateBoilTimeSliderValueFromProgress(this.seekbar_boiltime.getProgress())); Cognito.getAnalytics().getEventClient().recordEvent(syncEvent); } </pre> <p>Note that it is a protected override of the void function <code>RecordAnalytics()</code></p> <p>The implementation adds in various user-defined values such as the temperature selected by the user, and records it as a <code>MaxTemp</code> metric. These metrics are then recorded as an event in Mobile Analytics</p>

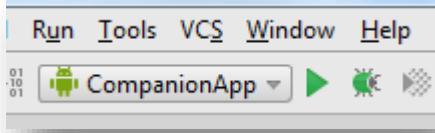
2.1.6

To implement this function, locate the file

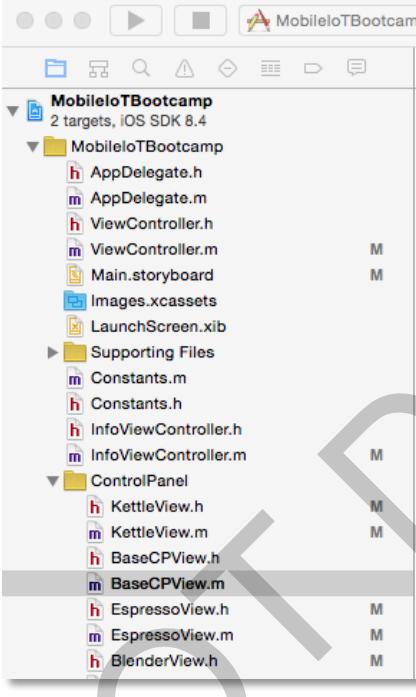
`Bootcamp2015-Android-Analytics-Kettle.txt`

in the Desktop bundle you downloaded in Lab 1.

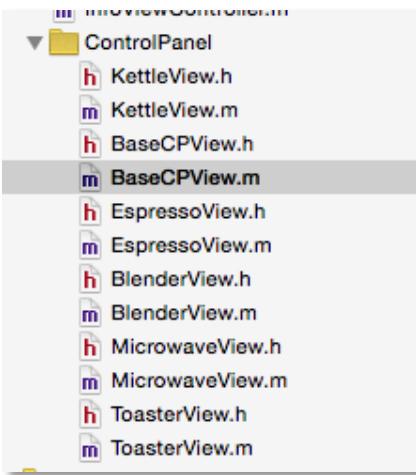
Paste the contents of this file into an appropriate location in the `KettleControlPanel` implementation file. For example, you could paste the function implementation just before the last curly brace in the file.

2.1.7	<p>When you are ready to start the build and test on your device, make sure CompanionApp is selected in the configuration list, and click the debug button (the bug icon) to start a build and debug session</p>  <p>The app will build and run as before (unless you made a mistake during the pasting in of the function!)</p> <p>Log in again (note that you don't need to re-tether) and in the Main Activity, touch the Control Panel icon (cogs icon) for your partner's appliance.</p> <p>The control panel will appear as before.</p>  <p>Make some changes to the settings, and the app will automatically synchronize with Cognito Sync, as before. When you are done, press the 'back' button on your Android device to return to the Main Activity</p> <p>When you return to the Main Activity, you will see a message notifying you that the analytics have been submitted successfully to Mobile Analytics</p> <p>It will take a few minutes for the analytics to appear on your Mobile Analytics dashboard, so we will come back to this in a moment</p>
2.1.8	

Task 2-1: Implement Mobile Analytics–Xcode

Step	Instruction
2.1.1	<p>In XCode, click on the Project Navigator in the upper left hand corner. Expand the MobileIoTBootcamp ControlPanel folder.</p> 

- 2.1.2 In the *controlpanels* folder you will see several Objective-C classes – one for each type of control panel our app supports, and a *BaseCPView*, from which all the other control panels derive.



Open the *BaseCPView.m* file by clicking on it.

2.1.3

Search for the method -(void)recordAnalytics: using CTRL+F

```
- (void) recordAnalytics{
    NSTimeInterval countSecondsPanelVisible = [[NSDate date] timeIntervalSinceDate:datePanelStarted];
    NSUserDefaults *prefs = [NSUserDefaults standardUserDefaults];
    AWSMobileAnalytics *analytics = [[AWSClientHelper sharedInstance] analytics:[prefs
        objectForKey:@"mobileanalyticsAppId"]];
    id<AWSMobileAnalyticsEventClient> eventClient = analytics.eventClient;
    id<AWSMobileAnalyticsEvent> panelEvent = [eventClient createEventWithEventType:[self
        getPanelNameForAnalytics]];
    [panelEvent addAttribute:@"identity" forKey:[[AWSClientHelper sharedInstance] getIdentityId]];
    [panelEvent addMetric:[NSNumber numberWithDouble:countSecondsPanelVisible] forKey:@"TimeVisible"];
    NSLog(@"Recording amount of time panel was open: %f",countSecondsPanelVisible);
    [eventClient recordEvent:panelEvent];
    [eventClient submitEvents];
}
```

The base class' implementation records the length of time the control panel was open for. The base class' implementation is called each time the Control Panel is closed. You can confirm this by looking at the –(void)viewWillDisappear: method in BaseCPView.

```
- (void) viewWillDisappear:(BOOL)animated{
    [super viewWillDisappear:NO];
    [self recordAnalytics];
}
```

When –(void)viewWillDisappear: is called by the iOS framework in the derived Control Panel class it will in turn call [super viewWillDisappear:NO] in BaseCPView. This happens automatically and no implementation in the derived class of –(void)viewWillDisappear: is needed.

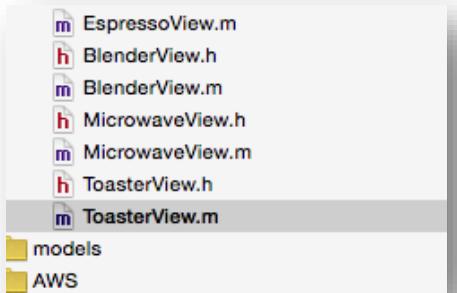
Your task is to implement Mobile Analytics custom events for your Control Panels.

2.1.4 If you have time, you can implement event capture for all the control panel types. However, we suggest you start with the appliance type that your partner has defined as their appliance type. The reason for this is, only your partner's appliance will show in your nearby appliance list, so if you want to be able to use the control panel, you will need to implement the control panel for their particular appliance.

Later, if time permits, you can pair with other students in the class, and so

may need to implement other control panels based on the appliance types they have chosen

Let's assume you need to implement the **Toaster** control panel.



Click on the *ToasterView.m* file in the solution explorer

2.1.5	<p>The code you need to implement is shown here:</p> <pre> - (IBAction)updateButton:(id)sender { AWSognito *syncClient = [AWSognito defaultCognito]; NSUserDefaults *preferences = [NSUserDefaults standardUserDefaults]; //Sync Cognito Dataset AWSognitoDataset *dataset = [syncClient openOrCreateDataset:[preferences objectForKey:@"uniqueDatasetName"]]; NSLog(@"Synchronizing dataset with Cognito"); [dataset synchronize]; //Send data to Mobile Analytics AWSMobileAnalytics *analytics = [[AWSClientHelper sharedInstance] analytics:[preferences objectForKey:@"mobileanalyticsAppId"]]; id<AWSMobileAnalyticsEventClient> eventClient = analytics.eventClient; id<AWSMobileAnalyticsEvent> customEvent = [eventClient createEventWithEventType:[self getPanelNameForAnalytics]]; [customEvent addMetric:[NSNumber numberWithFloat: [self calcTempValueSliderProgress; self.toastDarkness.value]] forKey:@"MaxTemp"]; [customEvent addMetric:[NSNumber numberWithFloat: [self calcTimeSliderProgress; self.toastWarmtime.value]] forKey:@"Time"]; [eventClient recordEvent:customEvent]; NSLog(@"Sending temperature and events to Mobile Analytics"); [eventClient submitEvents]; } </pre> <p>Note that this method is an IBAction that is already linked to the Storyboard so that dataset synchronization only happens when a user presses the 'Update' button. You modified this method in Lab 8 and are now adding logic for capturing Amazon Mobile Analytics custom events.</p>
2.1.6	<p>To implement this function, locate the <code>Bootcamp2015-iOS-ControlPanel-Toaster.txt</code> file in the Desktop bundle you downloaded in Lab 1.</p> <p>Locate the <code>- (IBAction)updateButton:</code> method towards the end of the file and paste the contents of the method implementation from this file into the method implementation in <code>KettleView.m</code>. <u>Do not copy and paste over the method or you may break the Storyboard connection.</u></p>

2.1.7	<p>When you are ready to start the build and test on your device, make sure iPhone is selected and press the arrow to the left.</p> 
	<p>The app will build and run as before (unless you made a mistake during the pasting in of the function)</p>
	<p>Log in again (note that you don't need to re-tether) and this time, in the Main View, touch the Control Panel icon (cogs icon) for your partner's appliance.</p>
	<p>The control panel will appear. Here is an example of the Toaster Control Panel.</p> 
	<p>Make some changes to the settings, and the app will automatically synchronize with Cognito Sync, as before. When you are done, press the 'back' button on your iOS device to return to the Main View</p>
2.1.8	<p>It will take a few minutes for the analytics to appear on your Mobile Analytics dashboard, so we will come back to this in a moment</p>

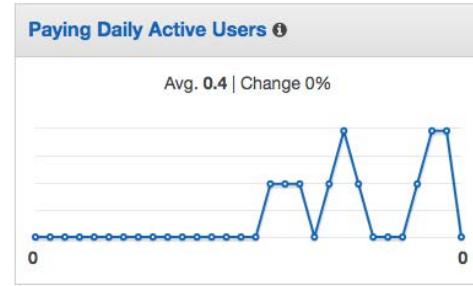
Task 3: Capturing In-app Purchase Analytics

Overview

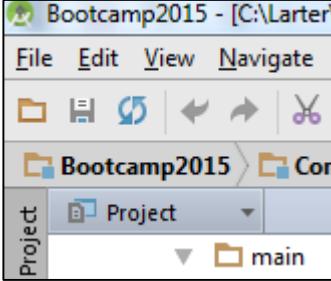
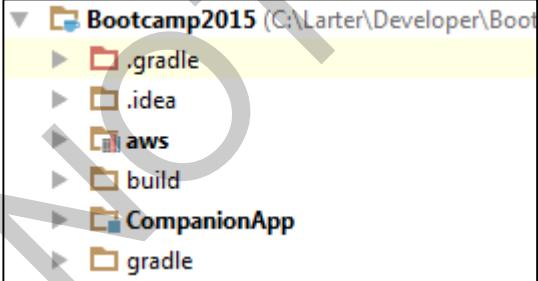
In order to demonstrate the analytics related to revenue generated by your mobile app, we use the monetization event feature provided by the SDK, to ‘charge’ the user each time one of the control panels is displayed. Of course, there is no charging, but the charge event is recorded against mobile analytics so that you can track the charge in the mobile analytics console.

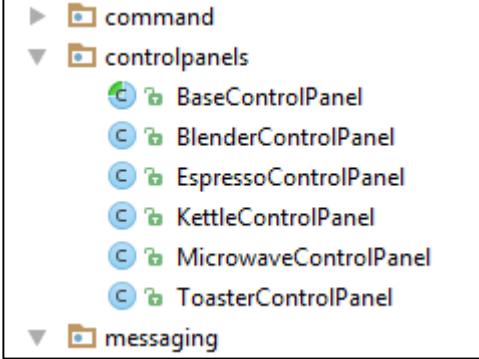
When the control panel is launched a charge event is recorded for a random (and dummy) SKU, for \$0.99 and the user is informed using a non-blocking alert (‘Toast’ in Android terminology)

Average Lifetime Value Per User ⓘ
\$59.07 USD iOS: \$84.15 USD Android: \$8.91 USD

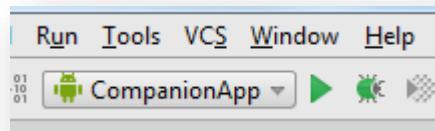


Task 3-1: Review the Charging for Control Panel Use implementation – Android Studio

Step	Instruction
3.1.1	<p>In Android Studio, ensure that the Project view is selected in the solution explorer panel on the left.</p>  <p>Locate the CompanionApp folder in the solution explorer panel, and expand it to reveal <i>CompanionApp src main java controlpanels</i></p> 

3.1.2	<p>In the <i>controlpanels</i> folder you will see several Java classes—one for each type of control panel our app supports, and a <i>BaseControlPanel</i>, from which all the other control panels derive</p>  <p>Open the <i>BaseControlPanel</i> by double clicking on it</p>
3.1.3	<p>Search for the function <i>ChargeForPanel ()</i> using CTRL+F</p> <pre data-bbox="331 998 1261 1474"> protected void ChargeForPanel(String sku, String price) { // get the event client from our MobileAnalyticsManager instance EventClient eventClient = Cognito.getAnalytics().getEventClient(); // create a builder that can record purchase events for Google Play IAP GooglePlayMonetizationEventBuilder builder = GooglePlayMonetizationEventBuilder.create(eventClient); // build the monetization event with the product id, formatted item price, // transaction id and quantity (Google IAP currently only supports a quantity of 1) AnalyticsEvent purchaseEvent = builder.withProductId(sku).withFormattedItemPrice(price) .withTransactionId(new Date().getTime() + "").withQuantity(1.0).build(); // record the monetization event in AWS mobile analytics eventClient.recordEvent(purchaseEvent); Toast.makeText(BaseControlPanel.this, "You have been charged \$" + price + " to use this panel", Toast.LENGTH_LONG).show(); } </pre> <p>The implementation randomly chooses an SKU (unit for sale) and records a charge for an In-App Purchase (IAP) against AWS Mobile Analytics.</p> <p>It is important to note that no actual charging takes place, we simply call the AWS Mobile Analytics API via the SDK, to register that a charge has occurred, in order for Mobile Analytics to keep some statistics for us</p>
3.1.4	<p>As an exercise, you may want to make changes to this implementation. For example, randomize the quantity or price. This is left as an exercise for the reader (or you may simply leave the implementation as it is).</p>

- 3.1.5 When you are ready to start the build and test on your device, make sure **CompanionApp** is selected in the configuration list, and click the **debug** button (the bug icon) to start a build and debug session



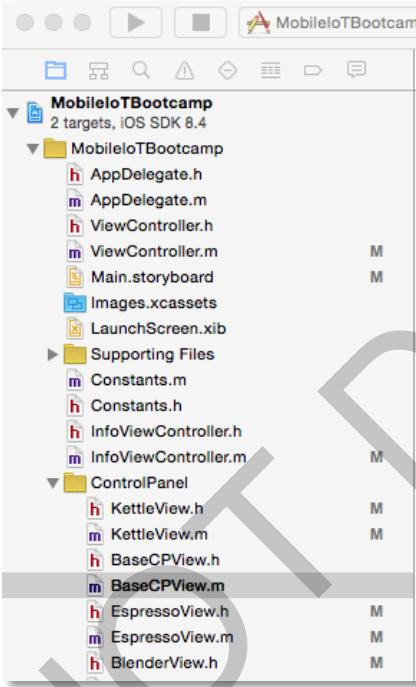
The app will build and run as before (unless you made a mistake if you edited the function).

Log in again (note that you don't need to re-tether) and this time, in the Main Activity, touch the Control Panel icon (cogs icon) for your partner's appliance.

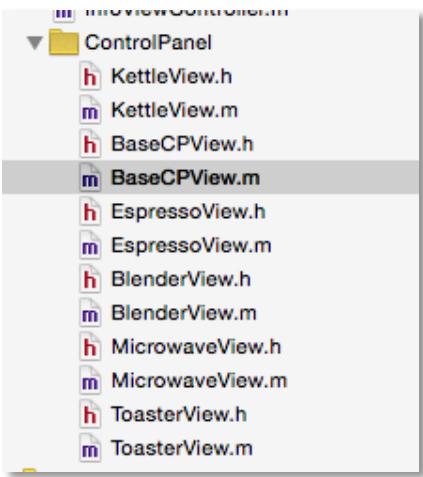
The control panel will appear as before. Adjust the values of the Control Panel, and press the 'back' button on the Android device. Do this a few times to create some statistics.

This will send analytics data to the AWS Mobile Analytics platform. In the next steps, we will browse to the custom events panel for your app in the console and review the data.

Task 3-1: Review Charging for Control Panel Use–XCode

Step	Instruction
3.1.1	In XCode, click on the Project Navigator in the upper left hand corner. Expand the MobileIoTBootcamp ControlPanel folder. 

- 3.1.2 In the *controlpanels* folder you will see several Objective-C classes – one for each type of control panel our app supports, and a *BaseCPView*, from which all the other control panels derive.



Open the *BaseCPView.m* file by clicking on it.

3.1.3

Search for the method -(void)chargeWithSku: using CTRL+F

```
- (void) chargeWithSku:(NSString*)sku andPrice:(double)price{
    NSLog(@"Recording monetization event in Mobile Analytics for sku: %@",sku);
    NSUserDefaults *prefs = [NSUserDefaults standardUserDefaults];
    AWSMobileAnalytics *analytics = [[AWSClientHelper sharedInstance] analytics:[prefs
        objectForKey:@"mobileanalyticsAppId"]];

    id<AWSMobileAnalyticsEventClient> eventClient = analytics.eventClient;
    AWSMobileAnalyticsAppleMonetizationEventBuilder *builder =
        [AWSMobileAnalyticsAppleMonetizationEventBuilder builderWithEventClient:eventClient];

    [builder withProductId:sku];
    NSLocale *usLocale = [[NSLocale alloc] initWithLocaleIdentifier:@"en_US"];
    [builder withItemPrice:price andPriceLocale:usLocale];
    [builder withQuantity:1.0];
    NSDateFormatter *dateFormat = [[NSDateFormatter alloc]init];
    [dateFormat setDateFormat:@"yyyy-MM-dd-hh:mm:ss"];
    [builder withTransactionId:[dateFormat stringFromDate:[NSDate date]]];
    [builder prepareForInterfaceBuilder];

    id<AWSMobileAnalyticsEvent> purchaseEvent = [builder build];
    [eventClient recordEvent:purchaseEvent];

    [eventClient submitEvents];
}
```

The implementation is called when the BaseCPView class's –(void)viewDidLoad: is invoked from an inherited derived class. It will pass a random SKU with a price of 0.99 cents. You can view this towards the top of BaseCPView.m

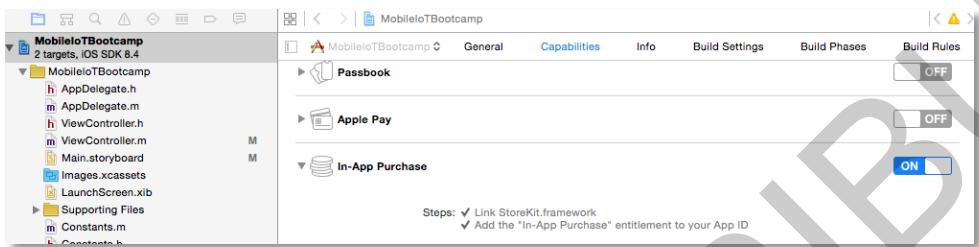
```
- (void) viewDidLoad{
    [super viewDidLoad];

    //SKUs for charging each time panel is launched
    NSArray *arraySKUs = [NSArray arrayWithObjects:@"SKU_1",
                           @"SKU_2",
                           @"SKU_3",
                           @"SKU_4",
                           @"SKU_5",
                           @"SKU_6",
                           @"SKU_7",
                           @"SKU_8",
                           @"SKU_9",
                           @"SKU_10", nil];

    int skuIndex = arc4random()%10;
    [self chargeWithSku:[arraySKUs objectAtIndex:skuIndex] andPrice:0.99];

    //Setup control panel image and title from tethered values
    NSUserDefaults *prefs = [NSUserDefaults standardUserDefaults];
    InfoViewController *info = [[InfoViewController alloc]init];
    //[[self.controlPanelImage setImage:[UIImage imageNamed:[info getApplianceImage:[prefs
        objectForKey:@"applianceType"]]]];
    NSMutableString *title = [NSMutableString string];
    [title appendString:[info getApplianceName:[prefs objectForKey:@"applianceType"]]];
    [title appendString:@" Control Panel"];
    self.controlPanelTitle.text = title;

    //Start recording how long control panel is open
    datePanelStarted = [NSDate date];
}
```

3.1.4	<p>In order to leverage in-app charging you should ensure that your XCode project is configured accordingly. Click MobileIoTBootcamp in the upper left-hand corner and select the Capabilities section. Ensure that In-App purchases are turned ON.</p>  <p>This should be sufficient for your purchases in Amazon Mobile Analytics. There is a corresponding setting in the Apple Developer Member Center for your App if you are creating a production application.</p>
3.1.5	<p>When you are ready to start the build and test on your device, make sure iPhone is selected and press the arrow to the left.</p>  <p>The app will build and run as before</p> <p>Log in again (note that you don't need to re-tether) and this time, in the Main View, touch the Control Panel icon (cogs icon) for your partner's appliance.</p> <p>The control panel will appear as before. Adjust the values of the Control Panel, and press the 'back' button on the iOS device. Do this a few times to create some statistics.</p> <p>This will send analytics data to the AWS Mobile Analytics platform. In the next steps, we will browse to the custom events panel for your app in the console and review the data.</p>

Task 3-2: View Mobile Analytics captured from Our Application

Step	Instruction																																
3.2.1	<p>3.2.1 Navigate to the Mobile Analytics dashboard under Mobile Services→Mobile Analytics.</p>  <p>The screenshot shows the AWS Cloud Services navigation bar. The 'Mobile Analytics' service is highlighted with a blue background and white text. Other services listed include All AWS Services, Compute, Storage & Content Delivery, Database, Networking, Administration & Security, Analytics, and Application Services. Each service has a corresponding icon and a brief description below it.</p> <table border="1"><thead><tr><th>All AWS Services</th><th>Cognito</th><th>Mobile Analytics</th><th>SNS</th></tr></thead><tbody><tr><td>Compute</td><td>Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices.</td><td>Amazon Mobile Analytics is a service that lets you easily collect, visualize, and understand app usage data at scale.</td><td>Amazon Simple Notification Service (SNS) lets you publish messages to subscribers or other applications.</td></tr><tr><th>Storage & Content Delivery</th><td></td><td></td><td></td></tr><tr><th>Database</th><td></td><td></td><td></td></tr><tr><th>Networking</th><td></td><td></td><td></td></tr><tr><th>Administration & Security</th><td></td><td></td><td></td></tr><tr><th>Analytics</th><td>AWS Device Farm helps you improve the quality of your Android, Fire OS, and iOS apps by testing them against real phones and tablets in the AWS Cloud.</td><td></td><td></td></tr><tr><th>Application Services</th><td></td><td></td><td></td></tr></tbody></table>	All AWS Services	Cognito	Mobile Analytics	SNS	Compute	Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices.	Amazon Mobile Analytics is a service that lets you easily collect, visualize, and understand app usage data at scale.	Amazon Simple Notification Service (SNS) lets you publish messages to subscribers or other applications.	Storage & Content Delivery				Database				Networking				Administration & Security				Analytics	AWS Device Farm helps you improve the quality of your Android, Fire OS, and iOS apps by testing them against real phones and tablets in the AWS Cloud.			Application Services			
All AWS Services	Cognito	Mobile Analytics	SNS																														
Compute	Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices.	Amazon Mobile Analytics is a service that lets you easily collect, visualize, and understand app usage data at scale.	Amazon Simple Notification Service (SNS) lets you publish messages to subscribers or other applications.																														
Storage & Content Delivery																																	
Database																																	
Networking																																	
Administration & Security																																	
Analytics	AWS Device Farm helps you improve the quality of your Android, Fire OS, and iOS apps by testing them against real phones and tablets in the AWS Cloud.																																
Application Services																																	

3.2.2

You will see some statistics from the use of your application



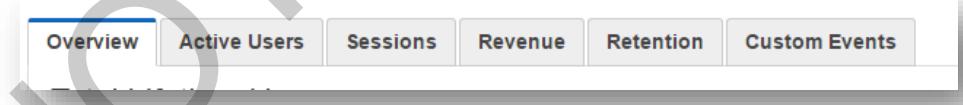
If you don't see any statistics, you may have to wait a few minutes for them to appear. In the meantime, confirm that you are indeed writing to the correct Mobile Analytics App Id:

- Browse to the Central website - <http://bit.ly/mobile-iot-bootcamp>
- Locate your Edison device in the list; copy the Appliance UUID into your clipboard
- Click the row that contains your appliance, to show the appliance detail window. You will see a view of the metadata that represents your Edison device:

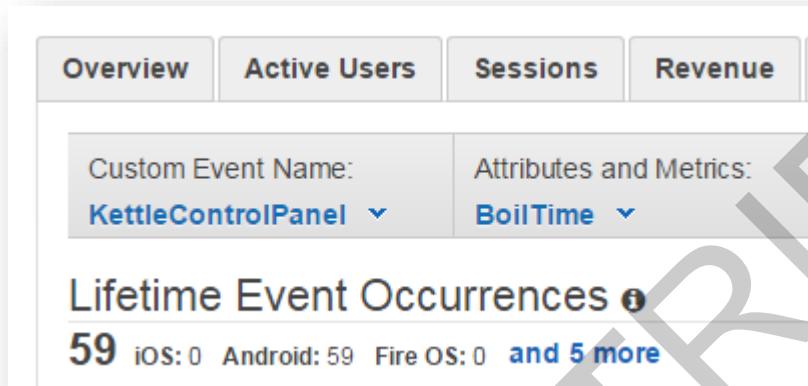
The screenshot shows the detail window for 'Adam's Edison'. It includes a red kettle icon and the identifier '984fee033fbc' (7 days ago). The metadata listed is:

Bluetooth UUID	badf00d5dff848d2b060d0f5a71096e0
IP Address	192.168.200.20
AWS Account ID	211983031378
Amazon SQS	https://sqs.us-east-1.amazonaws.com/211983031378/Bootcamp2015-Edison-Command
Identity Pool ID	us-east-1:1c3c8a68-d362-457f-ae43-3fae4e91d114
Mobile Analytics	82beb8f81e2144888e9893aedb8a471f
Cognito Auth Role	arn:aws:iam::211983031378:role/Bootcamp2015-Cognito-Authenticated
Cognito UnAuth Role	arn:aws:iam::211983031378:role/Bootcamp2015-Cognito-UnAuthenticated
SNS From Edison	arn:aws:sns:us-east-1:211983031378:Bootcamp2015-From-Edison
SNS To Mobile	arn:aws:sns:us-east-1:211983031378:Bootcamp2015-To-MobileApp

To the right of the metadata is a large QR code.

3.2.3	<p>Locate the value 'Mobile Analytics' entry and confirm it has a value that matches your Mobile Analytics App Id, as per task 1.1.8 on this Lab guide. If not, you may not have commanded your Edison to re-register after making a change to the /etc/bootcamp-config.js file in 1.1.9. Refer to that section to confirm.</p>  <p>If the value is correct, you may now re-tether your companion mobile app so that it picks up the change in the Mobile Analytics App Id you just set. You should re-tether now.</p>
3.2.4	<p>The detail in the Mobile Analytics dashboard for your app shows various statistics, including the average revenue for your users. Your data will be minimal because you only have one user and not much time to capture data, but this will give you an idea of what is possible with Amazon Mobile Analytics.</p>
3.2.5	<p>Click the Custom Events tab</p> 

- 3.2.6 Depending on the control panels you have implemented and tested, you will be able to select from the Custom Event Name drop-down list to locate the custom data you have captured

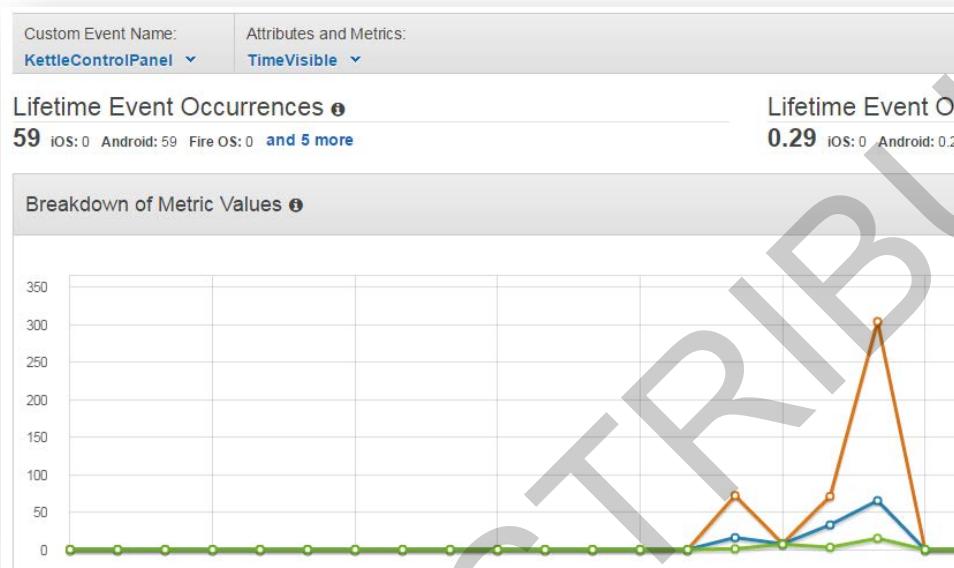


Here, we select the statistics for the custom event BoilTime



3.2.7

Here, the statistics for how long the panel has been visible is displayed.



Because you have not had a long time to capture the data, and the different control panels and users, your dataset will be limited. However, this gives you a good indication of how simple it is to create your own custom events and to capture attributes and metrics in Amazon Mobile Analytics.