



08.10.2017

Welcome to Audible! We hope you're enjoying your day so far. Here's what we have planned for you:

- 09:30 to 09:45 - Breakfast
- 09:45 to 10:30 - Introduction to Audible
- 10:30 to 11:15 - Panel discussion with Caroline Mwaura, Amie Greenwald, Akanksha Gupta, and Cynthia Chu
- 11:15 to 12:10 - Audible Studios and Newark Venture Partners tours
- 12:10 to 13:00 - Lunch & informal chats
- 13:00 to 15:00 - Engineering workshop
- 15:00 to 15:30 - Wrap-up

Now that you've heard from some of our leaders and toured our spaces, it's time for a little one-on-one with Alexa. 😊

During the engineering segment of today's field trip, you will be working in **groups of two** to build Amazon Alexa skills using Audible API. In the interest of time, we've provided skeleton code to get you started (you fill in the meat!).

The introduction contains background on AWS, developing on Alexa, and Audible API. This is followed by a series of short exercises that build on each other.

- Exercise One: Understanding the skeleton code
- Exercise Two: "Alexa, ask Story Teller to play me a random story"
- Exercise Three: Extending Story Teller with metadata
- Exercise Four: Be Creative

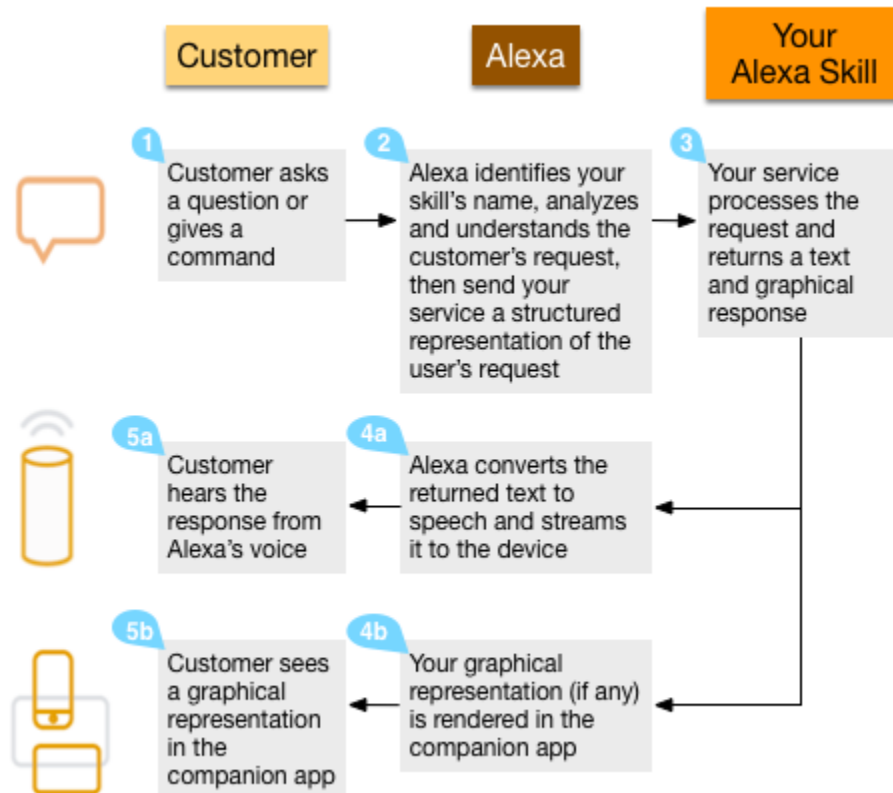
Please note that this document is meant to guide you through some activities and is meant for your personal use only. The last page contains any resources including images, source code, packages, and instructions that may be of use to you.

Have fun!

The #GWCAudible team

## Introduction

The goal of today's tech activity is to implement an Alexa skill. We will be using a set of technologies to accomplish this goal. Some of the technologies include Python, Amazon Web Services (AWS) Lambda, AWS CloudWatch, the Amazon Developer Portal, an Echo Simulator, and Echo Dots to demo our work.



## Exercise One: Understanding the skeleton code

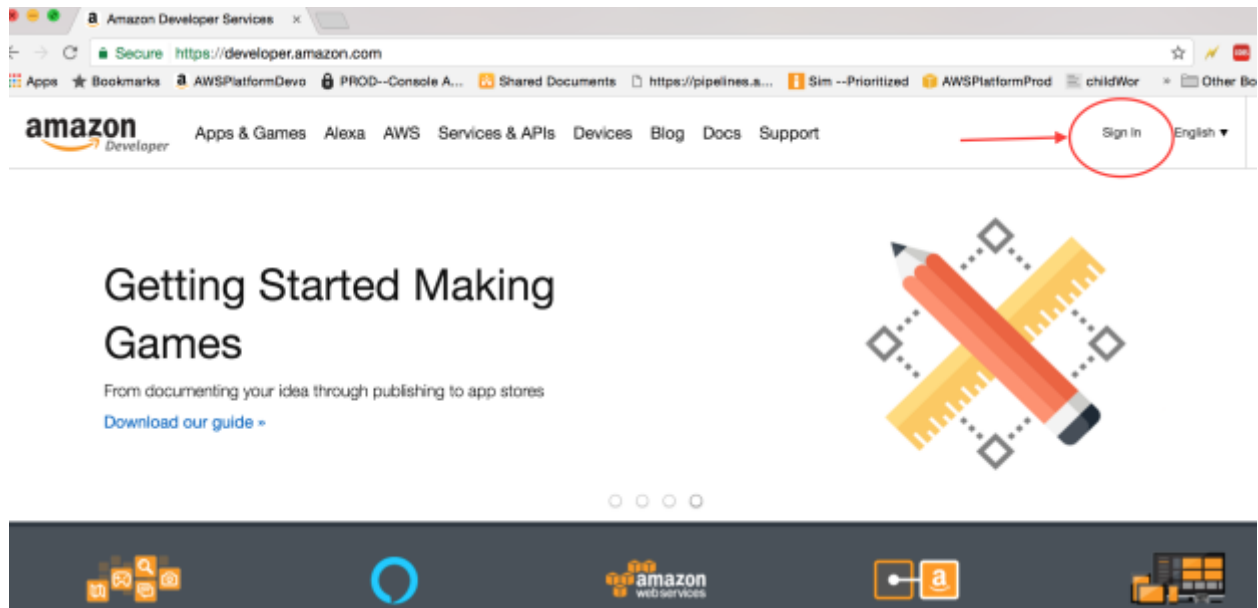
Alexa, ask Story Teller to tell me a story.

**Goal:** The goal of this activity is to set up your environment to interact with an Alexa simulator. You will configure your account so when you talk to the Alexa simulator your skill and ultimately your Python code will be executed. At the end of this activity, you should be able to ask Alexa to tell you a story.

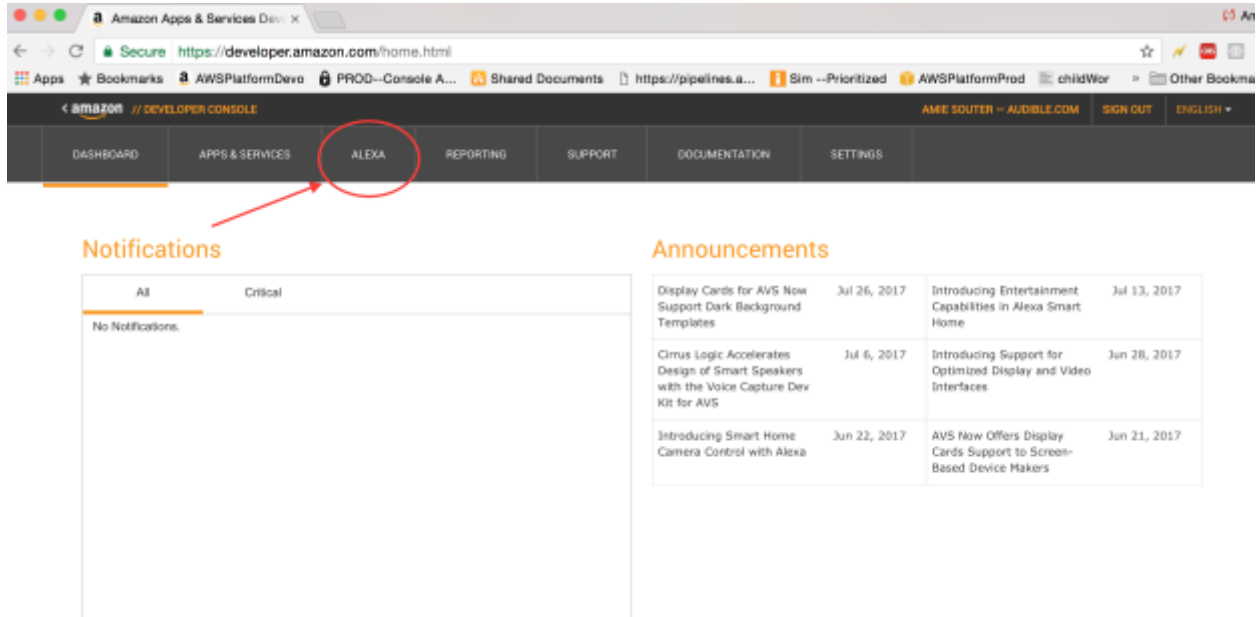
### Part 1: Creating an Alexa skill

Sign in to <https://developer.amazon.com>:

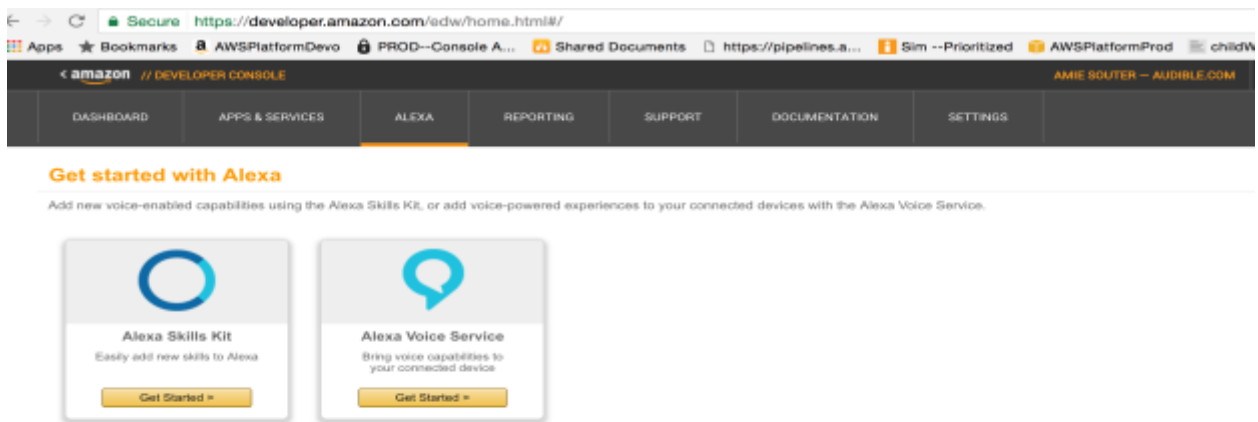
1. Go to the URL: <https://developer.amazon.com>
2. Click on the button Sign In on the top right corner.
3. Enter email and password. We will supply you with credentials.



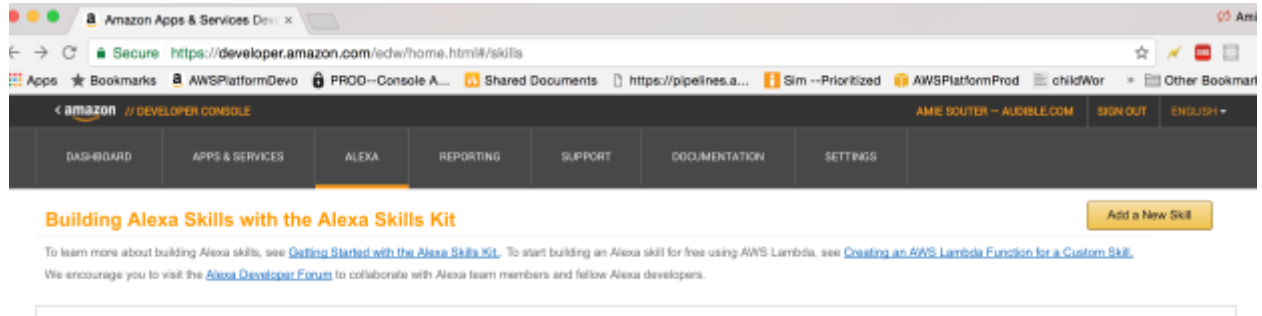
4. Once logged on, click on the “Alexa” tab.



5. Click on “Get Started” in the “Alexa Skills Kit” box.



6. Click on “Add a New Skill” (far right yellow button).



## 7. Skill Information

- A. **Skill Type:** Select “Custom Interaction Model”
- B. **Language:** Keep as English (for now)
- C. **Name:** Story Teller
- D. **Invocation Name:** Story Teller
- E. **Audio Player:** select “Yes”
- F. Click “Save”
- G. Click “Next”

A screenshot of the 'Create a New Alexa Skill' form in the Amazon Developer Console. The form is divided into several sections. On the left, there is a sidebar with a list of steps: Skill Information, Interaction Model, Configuration, SSL Certificate, Test, Publishing Information, and Privacy & Compliance, each with a checkmark. The main content area is titled 'Create a New Alexa Skill' and contains the following fields:

- Skill Type:** A dropdown menu with 'Custom Interaction Model' selected. Below it are three radio buttons: 'Smart Home Skill API', 'Flash Briefing Skill API', and 'Video Skill API'.
- Language:** A dropdown menu with 'English (U.S.)' selected.
- Name:** A text input field with 'Story Teller' entered.
- Invocation Name:** A text input field with 'Story Teller' entered.

Below these fields, there is a blue information box with a text icon and the text: 'For successful Alexa Skills Certification, please review and follow our [Invocation Name Guidelines](#) as well as our [Certification Requirements](#)'. At the bottom, there is a section titled 'Global Fields' with the text: 'These fields apply to all languages supported by the skill.' Below this, there is a field for 'Audio Player' with the text: 'Does this skill use the audio player directives?' and two radio buttons: 'Yes' (selected) and 'No'. There is also a 'Learn more' link.

## 8. Interaction Model

There are two boxes to fill in: **Intents** and **Sample Utterances**.

**Intents** map a user's voice input to services that your Alexa skill can address. More simply put, an intent represents an action that fulfills the end user's request. This mapping leverages a JSON structure called an intent schema.

Here we define three intents:

- 1) **StoryTeller**: when we encounter this intent, we can invoke the code in our Amazon Lambda function (we'll look at this in a bit).
- 2) **AMAZON.PauseIntent**: this is needed because we are going to be using the Audio Player and Alexa needs to know how to pause.
- 3) **AMAZON.ResumeIntent**: this is needed because we are going to be using the Audio Player and Alexa needs to know how to resume.

This is a really simple example—it's entirely possible to include more intents in your intent schema to allow your app do more things.

Add the following to **Intents**:

(link to file: <https://github.com/amieg/audiblegirlswhocode/blob/master/intents.json>)

```
{
  "intents": [
    {
      "intent": "StoryTeller"
    },
    {
      "intent": "AMAZON.PauseIntent"
    },
    {
      "intent": "AMAZON.ResumeIntent"
    }
  ]
}
```

The screenshot shows the Amazon Developer console interface. On the left, a sidebar contains navigation links: Skill Information (checked), Interaction Model (checked), Configuration (checked), Test (checked), Publishing Information (unchecked), and Privacy & Compliance (unchecked). Below this is a 'Skills Beta Testing' section with a 'Status: Not yet eligible' message. The main content area has a header with a 'Launch Skill Builder BETA' button. Below the header, the 'Intent Schema' section is visible, explaining that the schema is in JSON format and providing links to 'Intent Schema', 'built-in slots', and 'built-in intents'. A section titled 'Built-in intents for playback control' states that the skill uses the AudioPlayer directives and lists several built-in intents: AMAZON.ResumeIntent, AMAZON.LoopOnIntent, AMAZON.LoopOffIntent, AMAZON.ShuffleOnIntent, and AMAZON.ShuffleOffIntent. A button 'Add Pause/Resume intents' is present. Below this, a JSON snippet is shown in a code editor, defining the 'intents' array with 'StoryTeller', 'AMAZON.PauseIntent', and 'AMAZON.ResumeIntent'. At the bottom, the 'Custom Slot Types' section is partially visible, with a label 'Enter Type' and a text input field containing the word 'TYPE'.

After identifying the lambda function that Alexa will execute, we then need to link this to a human request that will trigger this linkage. To do this we leverage **utterances**.

Alexa leverages an organized and structured text file that maps intents to the skills by using a mapping file of likely utilized phrases called utterances.

Our app is really simple. In fact, to get it going, we've stripped out anything that might be asking for more than just triggering our lambda function. To trigger the lambda function, our intent looks for any of the following utterances. Each line of the utterances file always begins with the intent (in our case "StoryTeller") and then ends with what you might expect a user to ask (the utterance).

Add the following to **Sample Utterances** and **click** "Save" then hit "Next":



StoryTeller play me a story  
StoryTeller tell me a story

#### Sample Utterances

These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. [Learn more](#)

Up to 3 of these will be used as Example Phrases, which are hints to users.

- 1 StoryTeller play me a story
- 2 StoryTeller tell me a story

See [Certification Requirements](#) in our technical documentation as you develop your skills and prepare to submit to Amazon.

Save

Submit for Certification

Next

**Note: We will be skipping the Slots section.**

## 9. Configuration

1. AWS Lambda ARN (Amazon Resource Name)
2. North America
3. We will later enter the AWS Lambda ARN in the box under North America. For now, remember that this is an important piece we will revisit to allow Alexa to communicate with your Python code.

Secure [https://developer.amazon.com/edw/home.html#/skill/amzn1.ask.skill.3d3707e7-2540-4b7c-8572-65556b89e360/en\\_US/configure](https://developer.amazon.com/edw/home.html#/skill/amzn1.ask.skill.3d3707e7-2540-4b7c-8572-65556b89e360/en_US/configure)

Apps ★ Bookmarks AWSPlatformDevo PROD--Console A... Shared Documents https://pipelines.a... Sim --Prioritized AWSPlatformPro

English (U.S.) Add a New Language

**Skill Information** ✓  
**Interaction Model** ✓  
**Configuration** ✓  
**Test** ✓  
**Publishing Information** ✓  
**Privacy & Compliance** ✓

**Skills Beta Testing** NEW  
Status: Not yet eligible ⓘ

### Global Fields

These fields apply to all languages supported by the skill.

#### Endpoint

Service Endpoint Type: ☒ AWS Lambda ARN (Amazon Resource Name) ⓘ ☐ HTTPS

*Recommended*  
AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.  
[More info about AWS Lambda](#)  
[How to integrate AWS Lambda with Alexa](#)

Pick a geographical region that is closest to your target customers: ⓘ

☒ North America ☐ Europe

North America

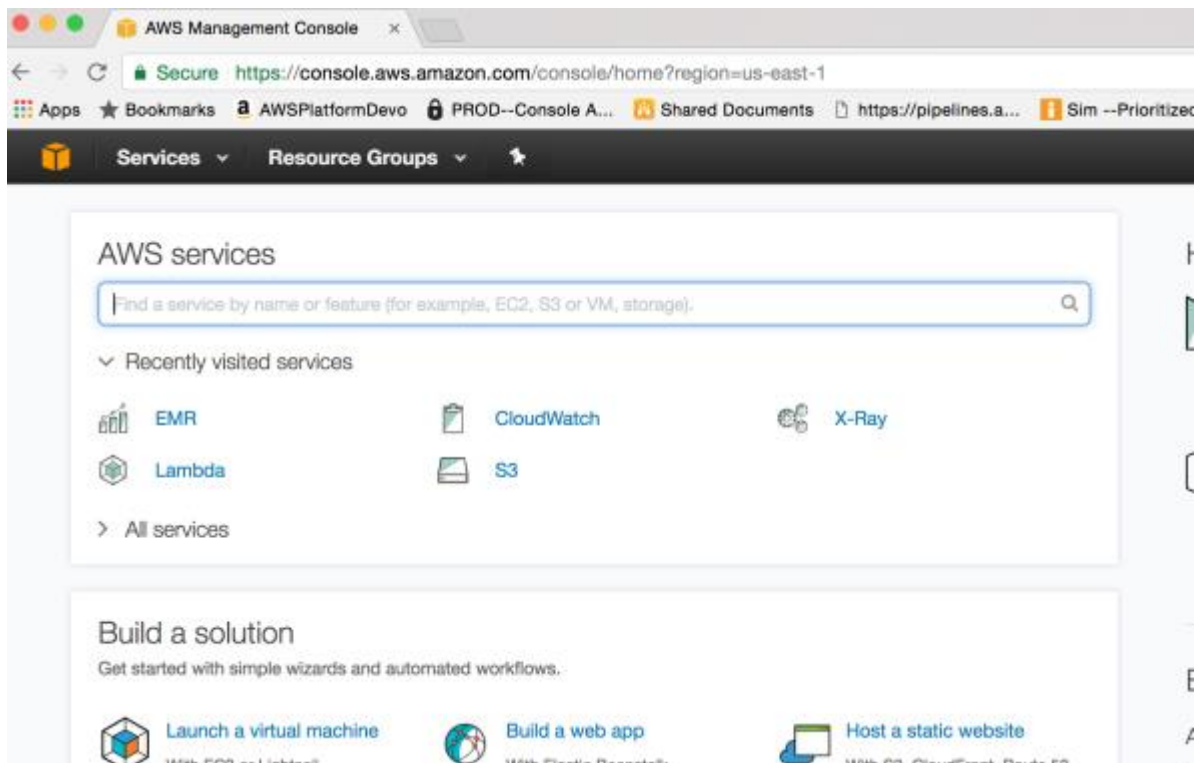
#### Account Linking

Do you allow users to create an account or link to an existing account with you? ☐ Yes ☒ No

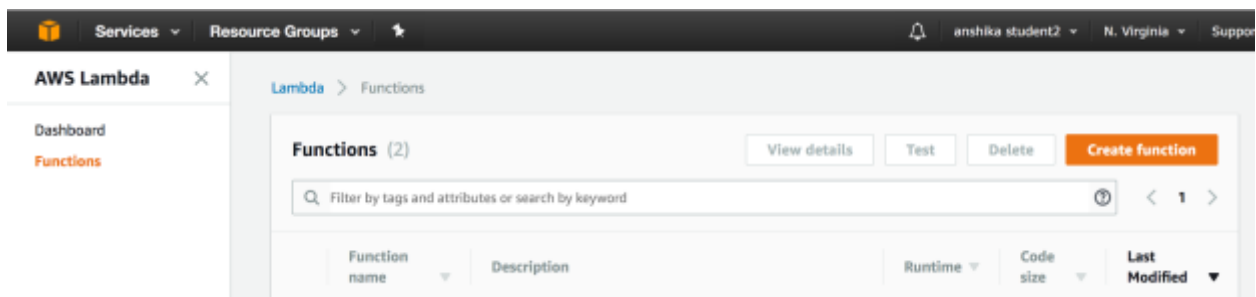
Part 2: **AWS Lambda**

Next, we are going to leave the developer portal and head over to Amazon Web Service (AWS). This is where we will write our Python code that will be executed when we talk to Alexa.

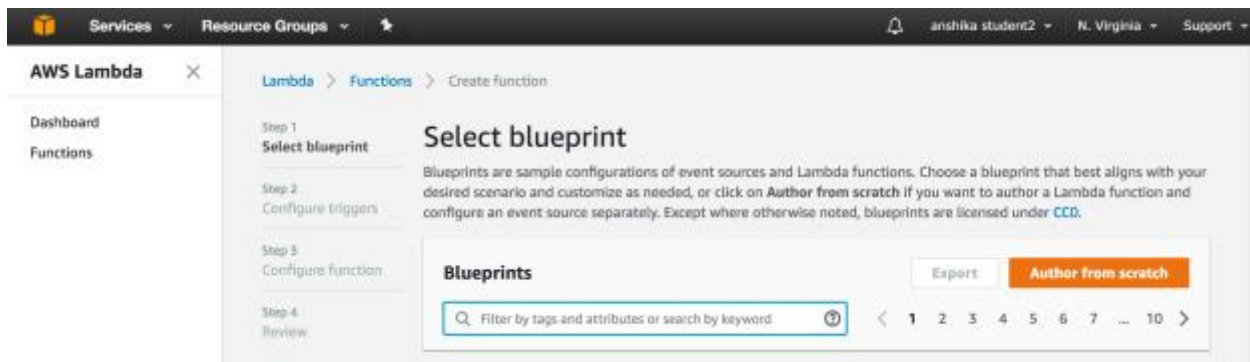
1. Sign into <https://aws.amazon.com> using the same email and password from the developer portal. Click the yellow button on the right that says “Sign In to the Console”
2. From the console, type “Lambda” in the search bar.



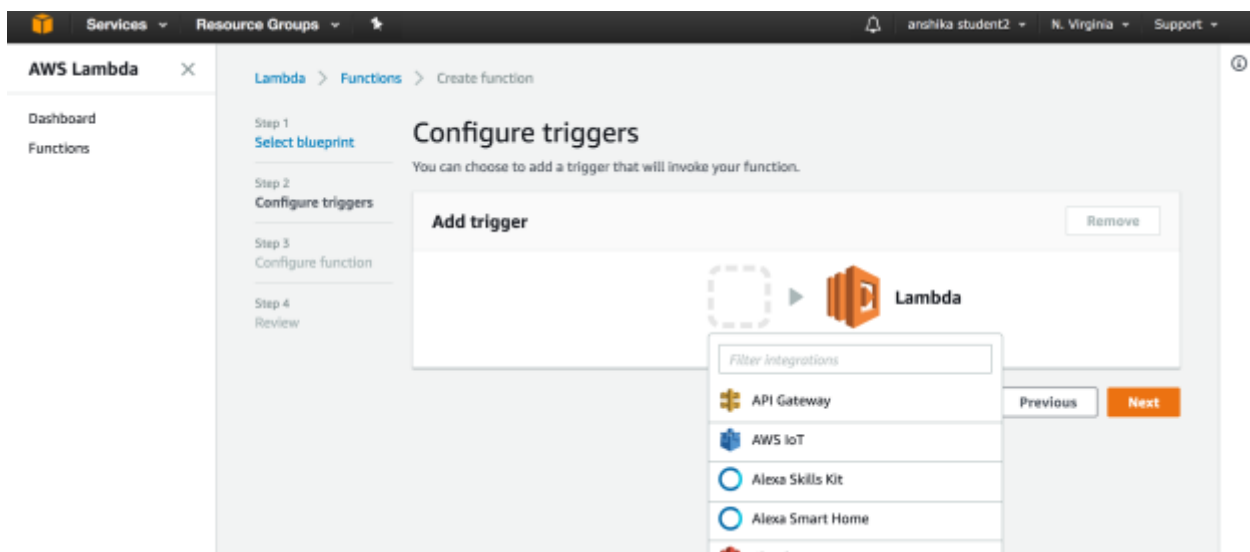
3. Click the Orange button that says “Create function”



4. Click the button “Author from scratch”



5. Configure triggers. Click on the dotted box and select “Alexa Skills Kit”



6. Click “Next”

7. This is where we will add our Python code. First, enter four more items:

**Name:** The name of your Lambda function.

**Description:** a description of the function.

**Runtime:** Python 2.7

**Lambda function code:** Copy the code from

<https://github.com/amieg/audiblegirlswhocode/blob/master/helloworld.py> and paste into the box. Make sure to delete the three lines that are already there.

Lambda Management Console

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

Apps Bookmarks AWSPlatformDevs PRD--Console A... Shared Documents https://pipelines.a... Sim --Prioritized AWSPlatformProd childWor

Lambda > New function

Select blueprint

Configure triggers

Configure function

Review

### Configure function

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

Name\* StoryTeller

Description My first Alexa skill.

Runtime\* Python 2.7

#### Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need out you can upload your code and libraries as a .ZIP file.

Code entry type Edit code inline

```

1- def lambda_handler(event, context):
2-     # TODO implement
3-     return 'Hello From Lambda'

```

8. Roles. Scroll down below the code box and enter role information as shown below.

### Lambda function handler and role

**Handler\***  
The filename.handler-method value in your function. For example, "main.handler" would call the handler method defined in main.py.

lambda\_function.lambda\_handler

**Role\***  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s)

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

**Role name\***  
Enter a name for your new role.

AWSLambdaBasicExecutionRole

**Policy templates**  
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

9. Click “Next”
10. Click the orange button “Create function”
11. We are almost done! An ARN will appear on the top right corner of the page. Copy the ARN - **arn:aws:lambda:us-east-1:73697XXXXXX:function:StoryTeller** (copy the part that starts with “arn:aws” to the end of the line).
12. Go back to the developer portal page and paste the ARN into the Configuration page in the box under “North America”
13. Click “Next”

### Part 3: Testing

Now we can test that everything is hooked up together and works properly. There are two ways we can test. The first method is to use the next Tab in the developer.amazon.com portal. The Service Simulator will allow us to type utterances and see the actual JSON lambda response. You can also listen to the response. See the figure below for an example.

1. Type an utterance in the text box.
2. Click the button “Story Teller”
3. Click the button next to “Listen” to hear the response.

The screenshot shows the AWS Lambda Service Simulator interface. At the top, there are two tabs: 'Text' (selected) and 'JSON'. Below the tabs is a text input field labeled 'Enter Utterance' containing the text 'play me a story'. Below the input field are two buttons: 'Ask Audible Story Teller' and 'Cancel'. Below these buttons are two sections: 'Lambda Request' and 'Lambda Response'. The 'Lambda Request' section displays a JSON object with the following structure:

```

{
  "session": {
    "sessionId": "fa20ac85-293f-4f03-
    "application": {
      "applicationId": "amzn1.ask.skill.fa8826e
    },
    "attributes": {},
    "user": {
      "userId": "amzn1.ask.account.AGTFTHYFVGJH
    },
    "new": true
  },
  "request": {
    "type": "IntentRequest",
    "requestId": "EdwRequestId.2ed6641b-9153-44
    "locale": "en-US",
  }
}

```

The 'Lambda Response' section displays a JSON object with the following structure:

```

{
  "requested": "<break time='1s' /> <audio src='http
"
}

```

At the bottom right of the interface is a 'Listen' button with a play icon. Red arrows point to the 'Enter Utterance' field, the 'Ask Audible Story Teller' button, and the 'Listen' button.

### Part 4: Using echosim.io

Try out your Alexa skill with echosim.io.

1. Go to <https://echosim.io>
2. Enter the same username and password you have been using.

3. Click the button and ask Alexa to play your skill (i.e. Alexa, ask Story Teller to tell me a story).

## Exercise Two: Alexa, ask Story Teller to play me a random story

For this activity, we are going to have Alexa read a random story. We will utilize the same code as before, this time calling a new API that gets Audible samples instead of playing the samples from the icebreakers.

1. Go to <https://www.audible.com> and search for six of your favorite books. If you don't have favorite books, just pick any six books. Go to the detail page of the book and copy the ASIN to a file. The ASIN appears in the URL after the title of the book. For example, **B074CMMWCW** is the ASIN associated with URL below.

[https://www.audible.com/pd/Bios-Memoirs/Happiness-A-Memoir-Audiobook/B074CMMWCW/ref=a\\_hp\\_c7\\_1\\_1\\_i\\_NINF\\_629?ie=UTF8&pf\\_rd\\_r=1EZFWAW55QNT99G8ESN9&pf\\_rd\\_m=A2ZO8JX97D5MN9&pf\\_rd\\_t=101&pf\\_rd\\_i=5000&pf\\_rd\\_p=3154417622&pf\\_rd\\_s=center-7](https://www.audible.com/pd/Bios-Memoirs/Happiness-A-Memoir-Audiobook/B074CMMWCW/ref=a_hp_c7_1_1_i_NINF_629?ie=UTF8&pf_rd_r=1EZFWAW55QNT99G8ESN9&pf_rd_m=A2ZO8JX97D5MN9&pf_rd_t=101&pf_rd_i=5000&pf_rd_p=3154417622&pf_rd_s=center-7)

2. In your favorite python editor, write a function that will randomly return one of the six random ASINs from Step 1.

If you are unsure how to get started, the following function might be useful, which returns a random number between 0 and 9. If you are still unsure, ask a volunteer for help.

```
from random import randint
randint(0,9)
```

2. Go to the lambda editor in the AWS console (<https://console.aws.amazon.com/>).
3. Paste the new function into lambda function.
4. Call the new function that you wrote so it will be a parameter to the function `getSample(...)`.
5. Save the lambda function.
6. Go to [developer.amazon.com](https://developer.amazon.com) and add to the utterances to play a random story.
7. Go to [developer.amazon.com](https://developer.amazon.com) and test out the changes to the skill.
8. Try it out on [echoism.io](https://echoism.io) too.

## Exercise Three: Extending Story Teller with metadata

For this activity, you will extend Story Teller by having Alexa provide the title, author, narrator of the story. For example,

**Request:** Alexa, ask Story Teller to tell me a story.

**Response:** This story is from <TITLE> written by <AUTHOR> and narrated by: <NARRATOR>. Here is a sample of the story:

1. Go to the lambda editor.

2. Call the function X
3. Add the output of the function call above to the output that is sent to Alexa.
4. Go to [developer.amazon.com](https://developer.amazon.com) and add to the utterances to play a random story.
5. Go to [developer.amazon.com](https://developer.amazon.com) and test out the changes to the skill.
6. Try it out on [echoism.io](https://echoism.io) too.

## Exercise Four: Be creative

If you have gotten to this section, great job! The Alexa API is large and we have only touched a small surface in terms of what is possible with Alexa. Please take this time to explore what else is possible with Alexa. One area that might be interesting to look at are Slots, which allow a user to provide information to Alexa that can be used as parameters.

## References:

- <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/alexa-skills-kit-interaction-model-reference>

- <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/built-in-intent-ref/slot-type-reference>
- <https://developer.amazon.com/public/solutions/alexa/alexa-voice-service/reference/audioplayer-conceptual-overview>
- <https://developer.amazon.com/blogs/tag/alexa>
- <https://developer.amazon.com/blogs/alexa/>
- <https://lovemyecho.com/>

## Wrap-up

Follow us and share your experiences using the hashtag #GWCAudible. You can look forwards to a blog post with some pictures on Signal to Noise (<http://stn.audible.com/>), our Design & Engineering blog. You can also follow us on these social channels:



[@audible\\_com](https://twitter.com/audible_com)



[@audible](https://www.facebook.com/audible)



[@audible\\_com](https://www.instagram.com/audible_com)