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

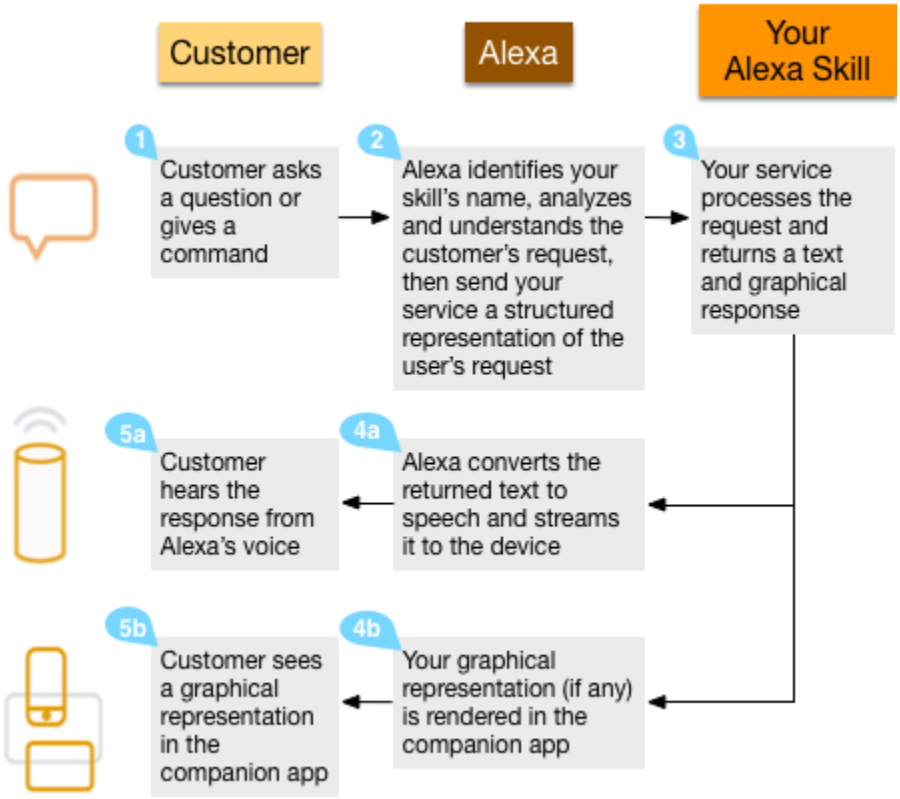
Please note that this document is meant to guide you through some activities and is meant for your personal use only. The last page contains any resources including images, source code, packages, and instructions that may be of use to you.

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

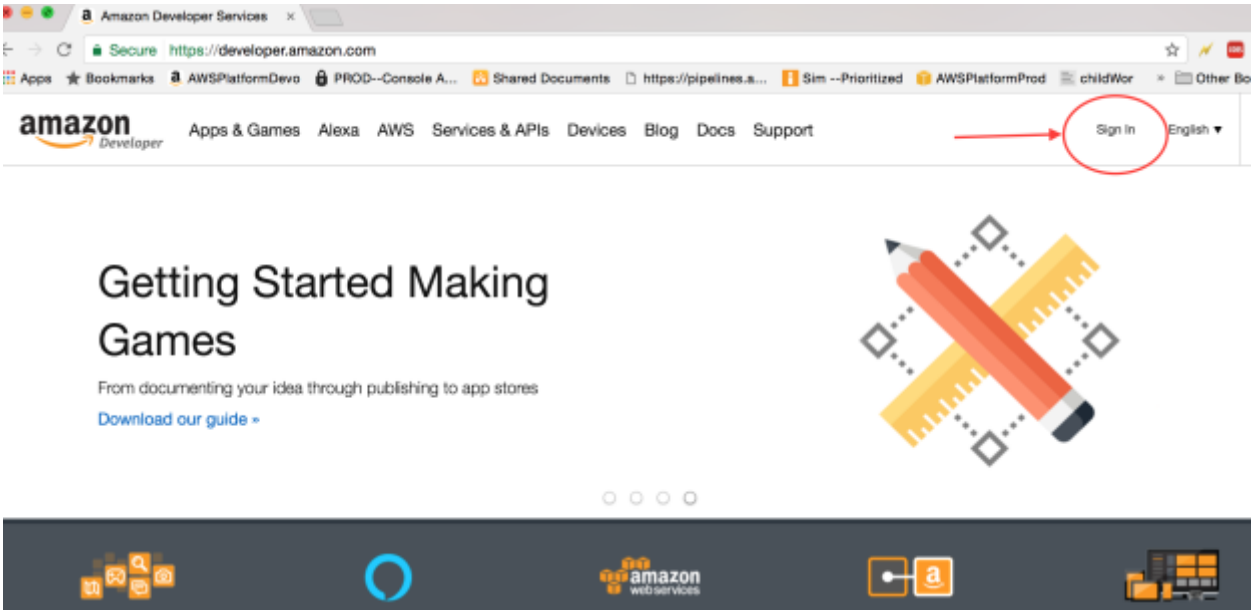
Alexa, ask Story Teller to tell me a story.

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.

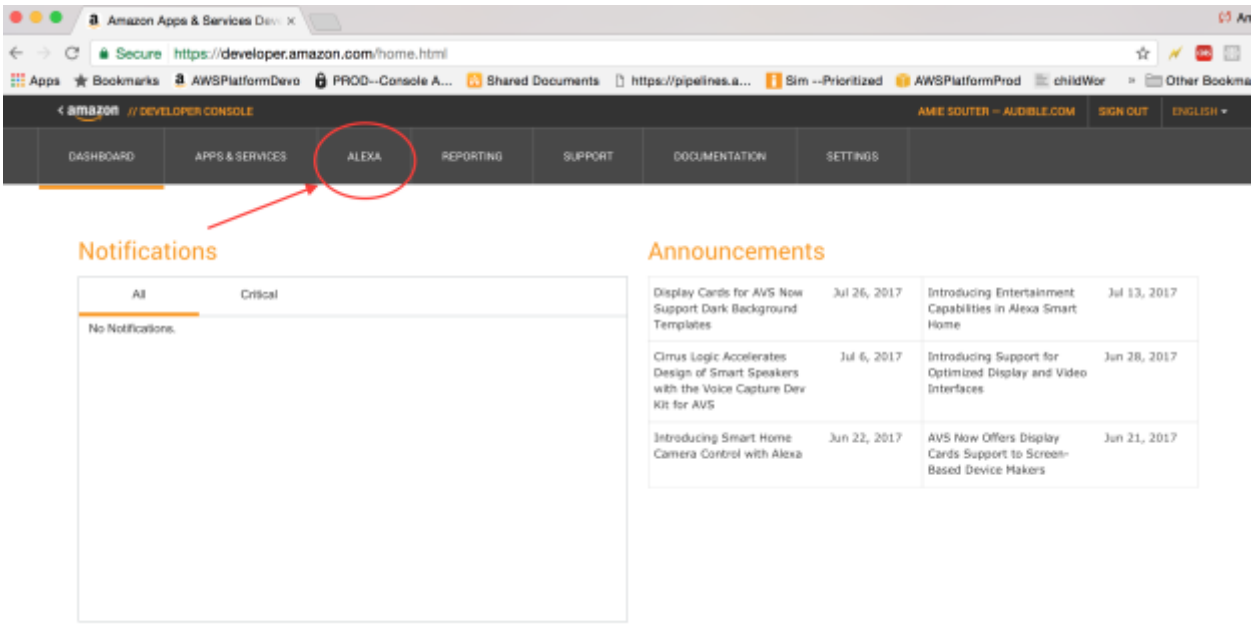
Part 1: Creating an Alexa skill

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

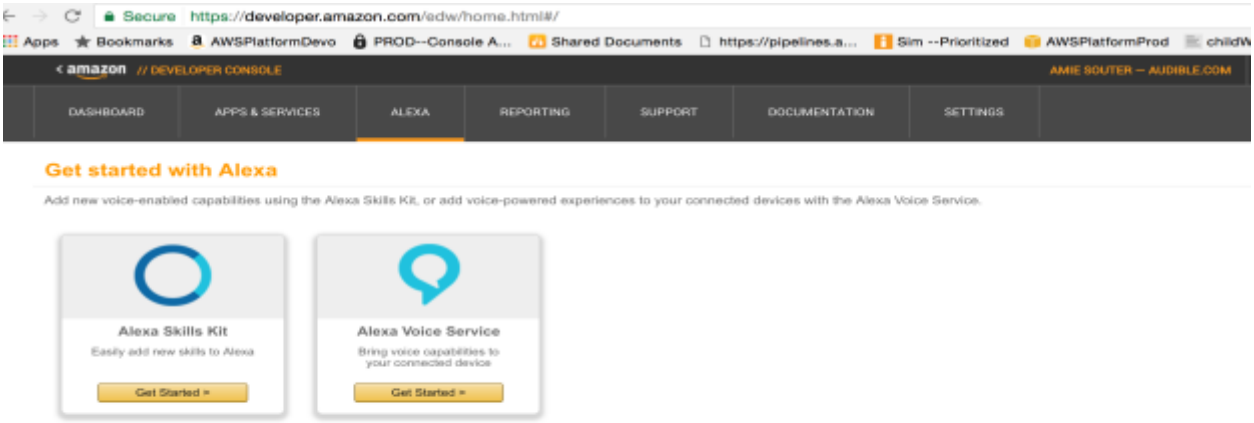
1. Go to the URL: https://developer.amazon.com
2. Click on the button Sign In on the top right corner.
3. Enter email and password. We will supply you with credentials.



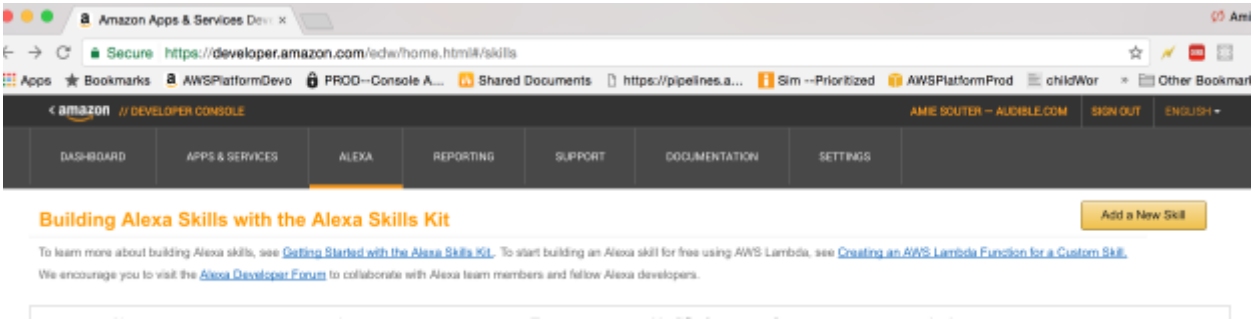
4. Once logged on, click on the “Alexa” tab.



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

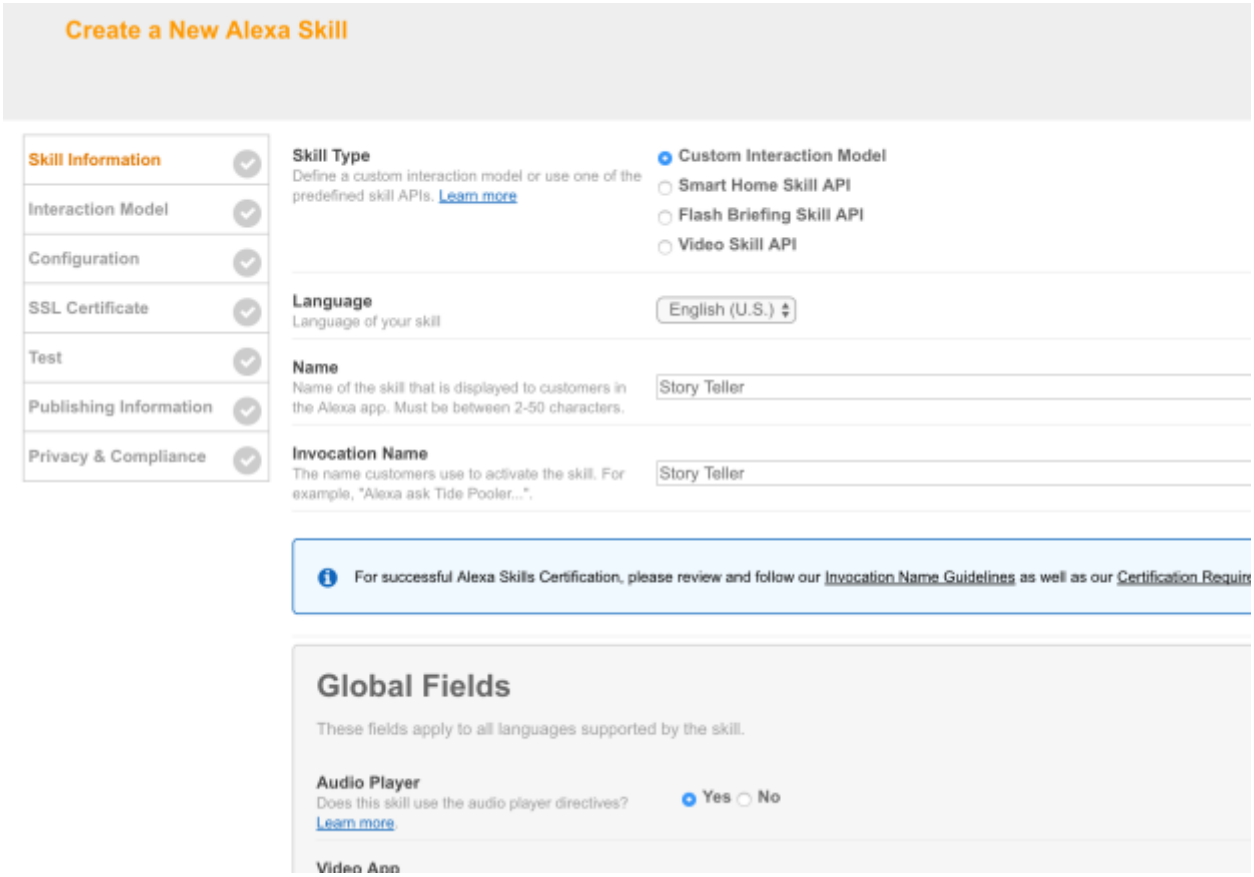


6. Click on “Add a New Skill” (far right yellow button).



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”



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.

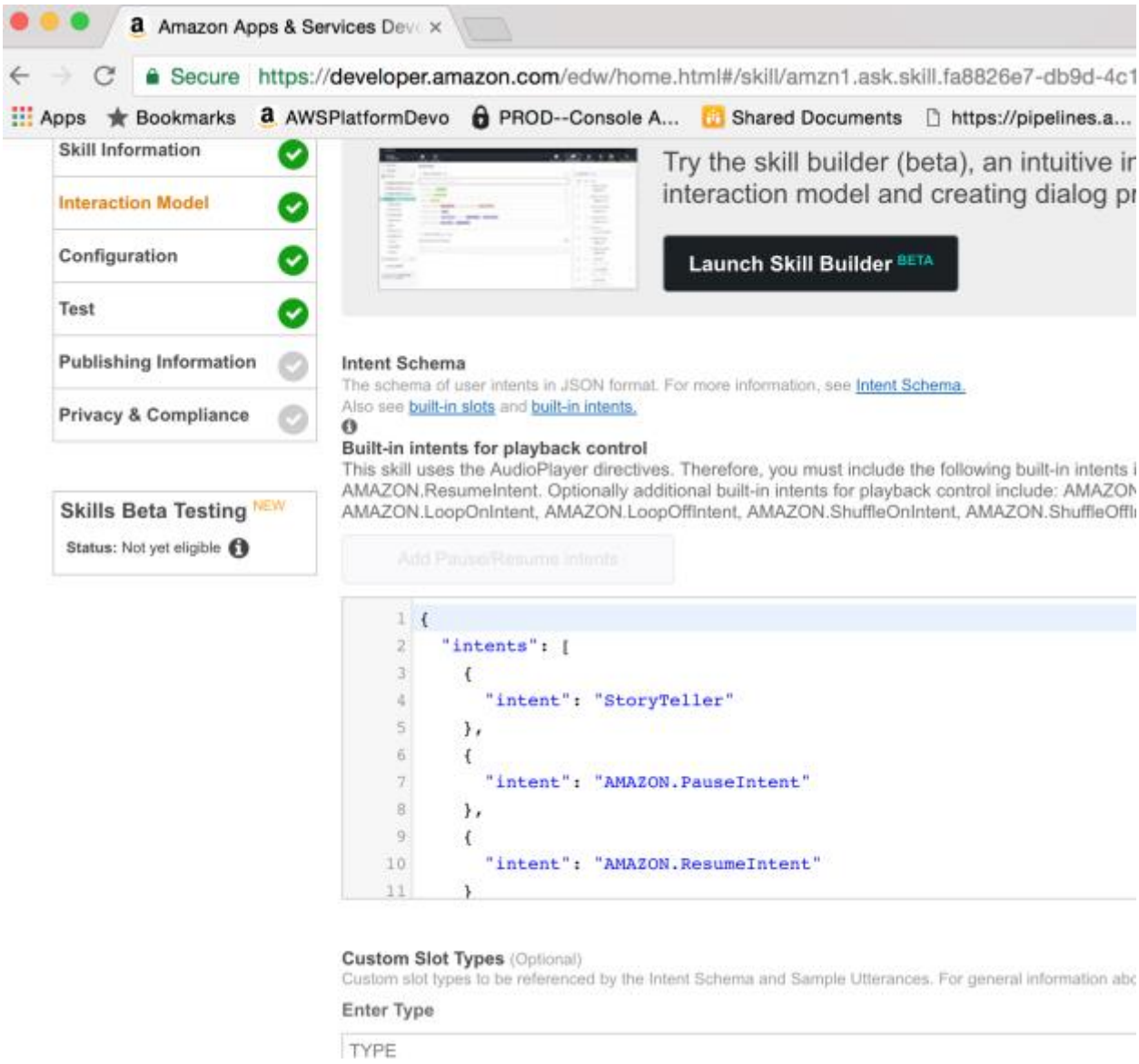
This is a really simple example–it's entirely possible to include more intents in your intent schema to allow your app do more things.

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"
    }

  ]
}
```



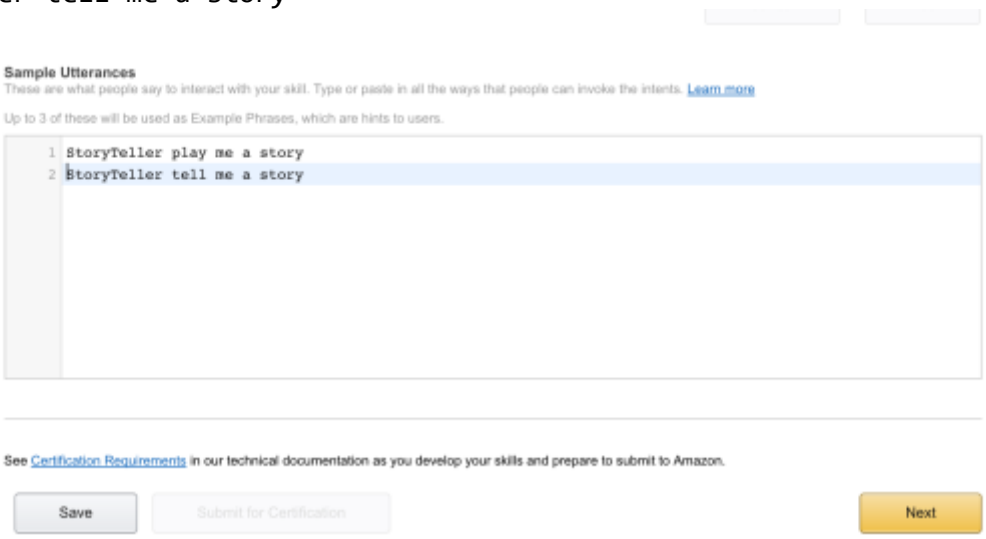
After identifying the lambda function that Alexa will execute, we then need to link this to a human request that will trigger this linkage. To do this we leverage **utterances**.

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).

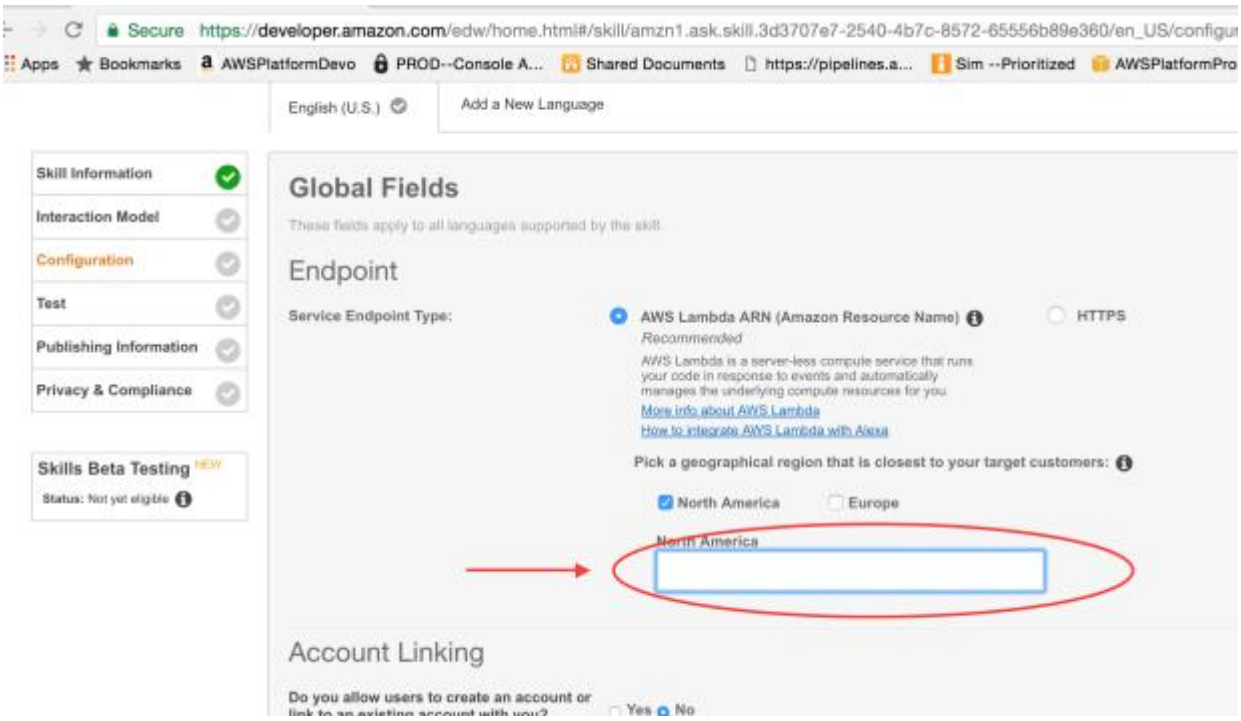
Add the following to **Sample Utterances** and **click** "Save" then hit "Next":

```
StoryTeller play me a story
StoryTeller tell me a story
```



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.

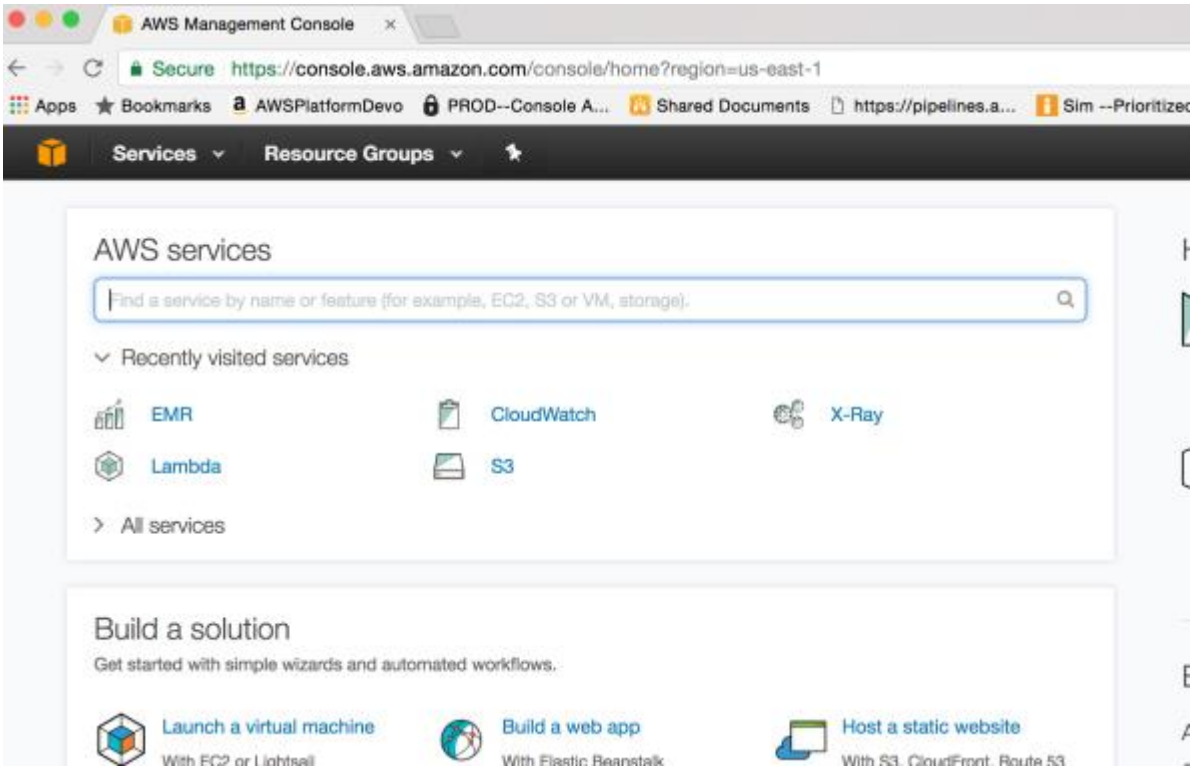


Part 2: **AWS Lambda**

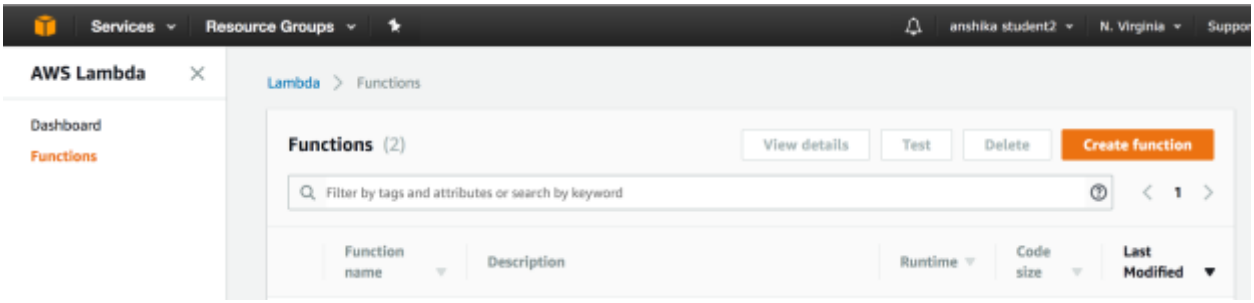
Next, we are going to leave the developer portal and head over to Amazon Web Service (AWS). This is where we will write our Python code that will be executed when we talk to Alexa.

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”

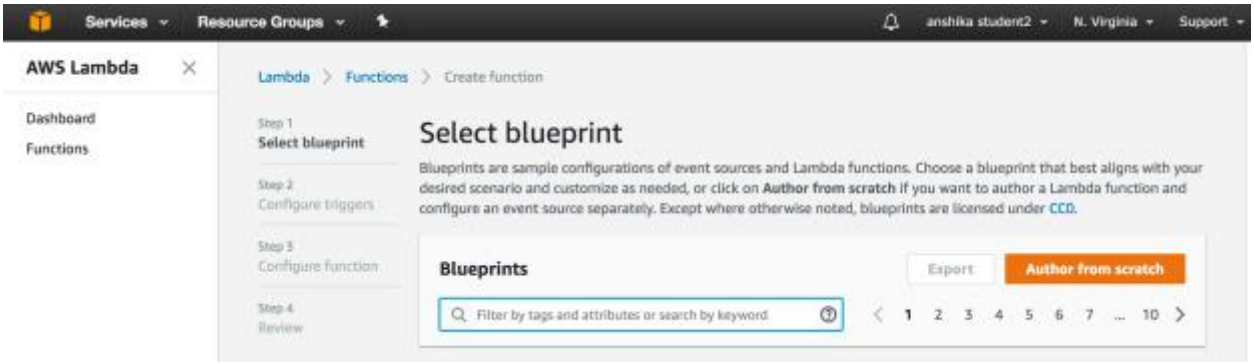
2. From the console, type “Lambda” in the search bar.



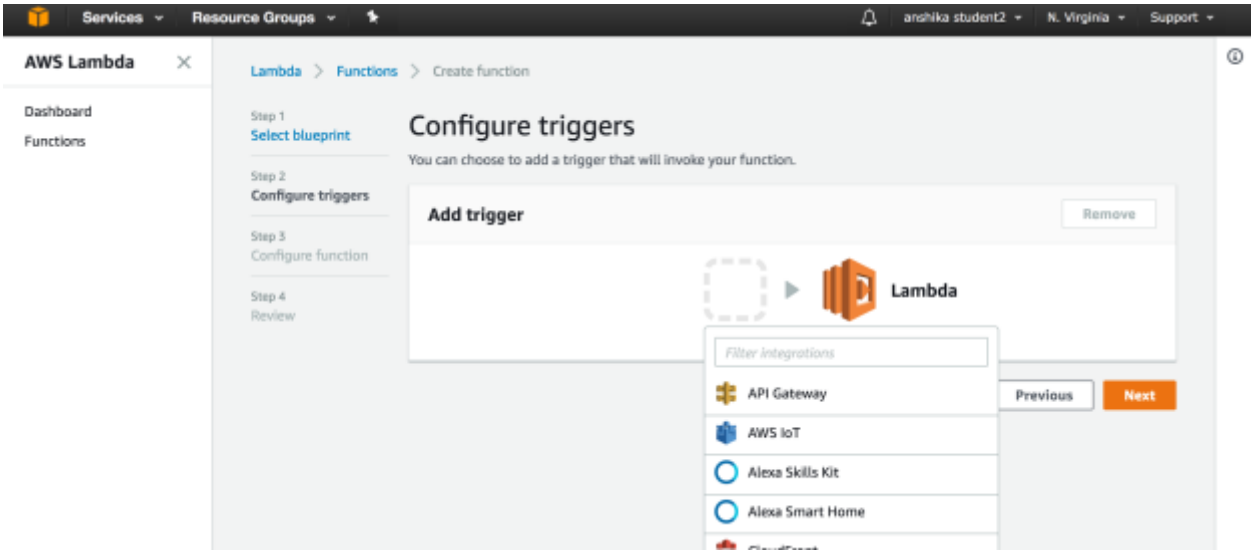
3. Click the Orange button that says “Create function”



4. Click the button “Author from scratch”

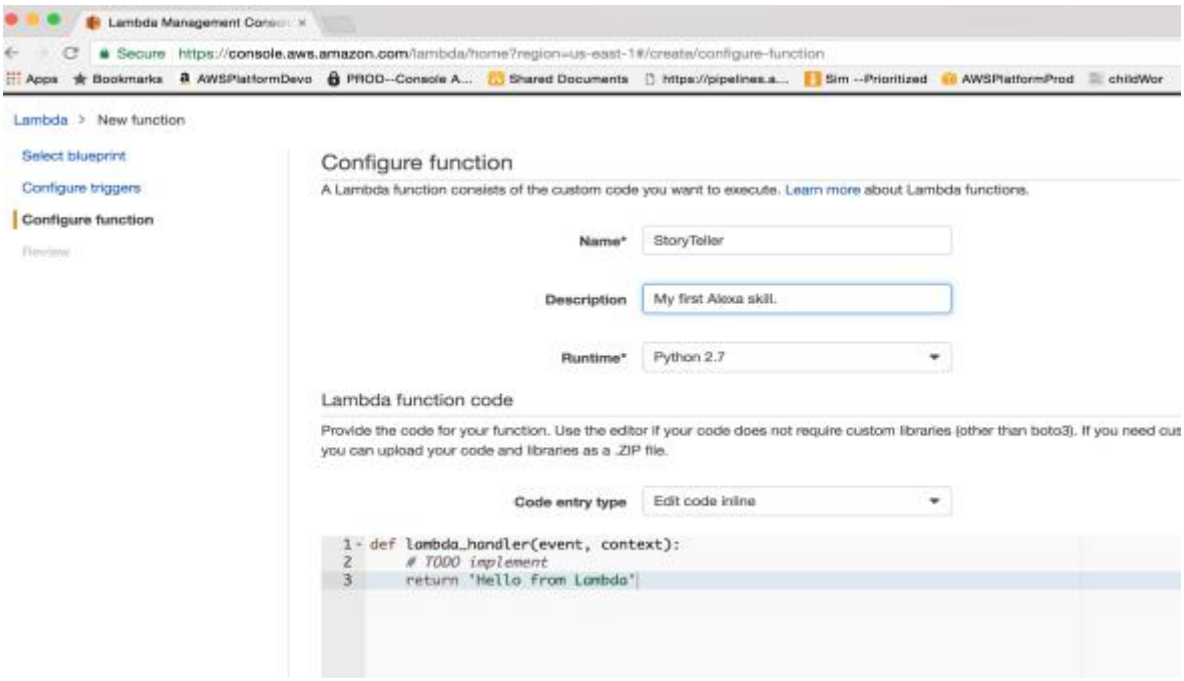


5. Configure triggers. Click on the dotted box and select “Alexa Skills Kit”

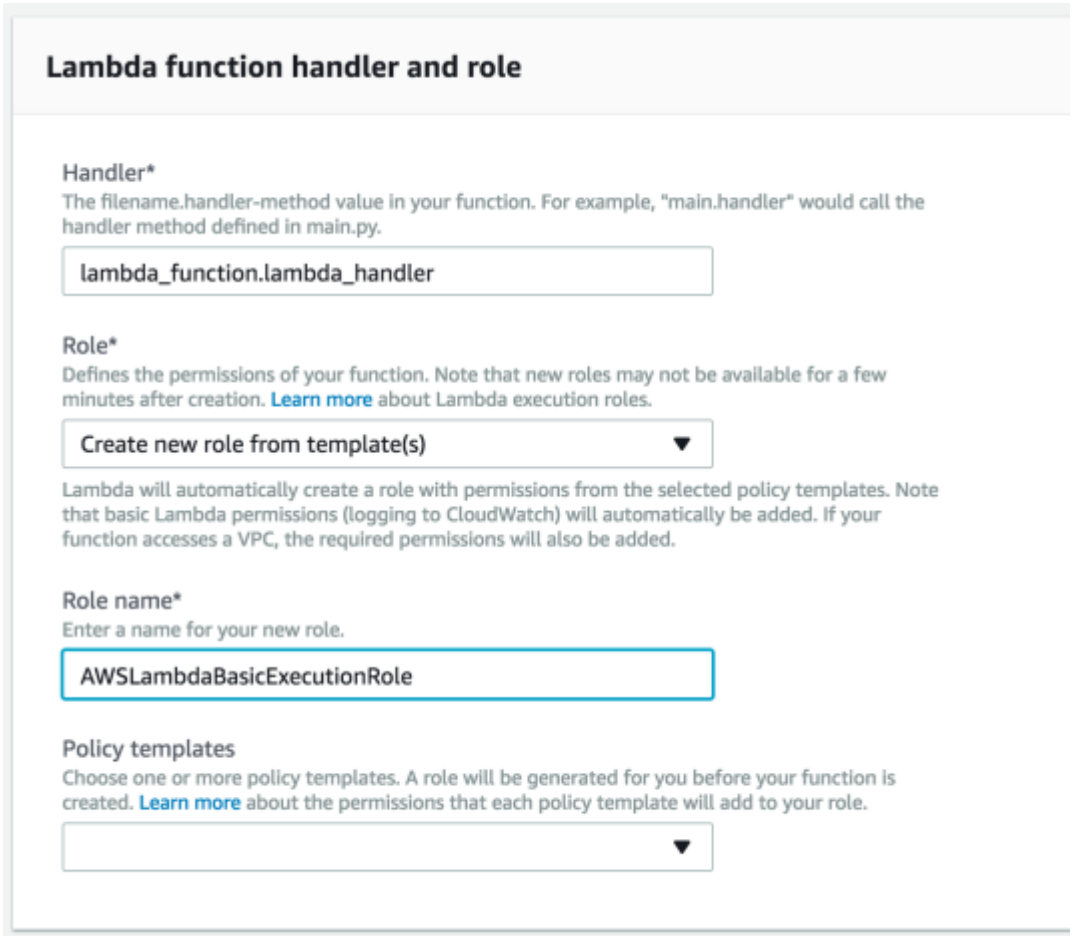


6. Click “Next”

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.



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



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-1:73697XXXXXXX:function:StoryTeller** (copy the part that starts with “arn:aws” to the end of the line).

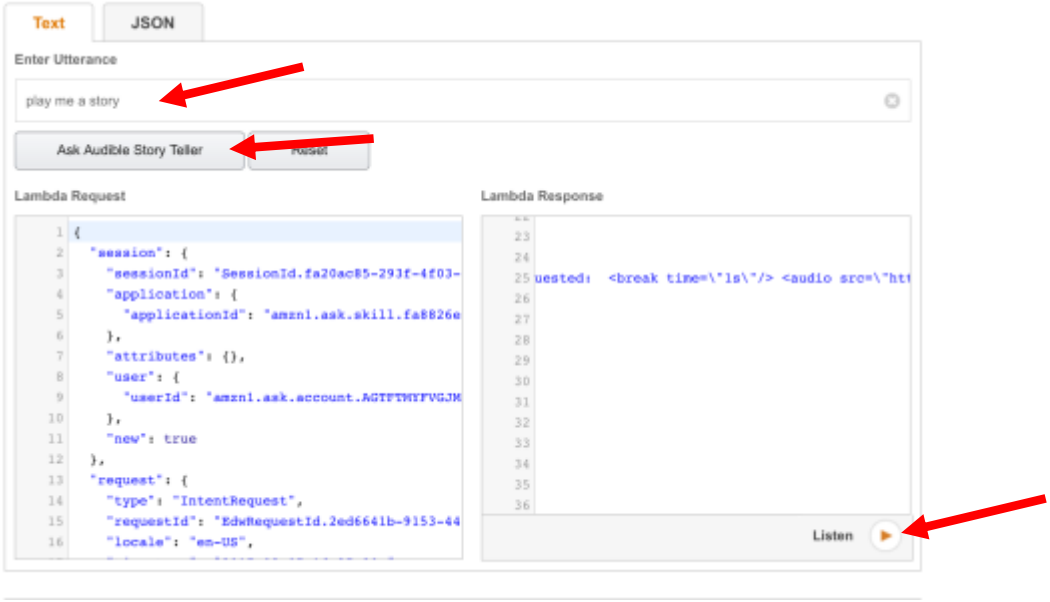
12. Go back to the developer portal page and paste the ARN into the Configuration page in the box under “North America”

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.



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

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 and add to the utterances to play a random story.
7. Go to developer.amazon.com and test out the changes to the skill.
8. Try it out on 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 and add to the utterances to playa a random story.
5. Go to developer.amazon.com and test out the changes to the skill.
6. Try it out on 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

 @audible

 @audible_com