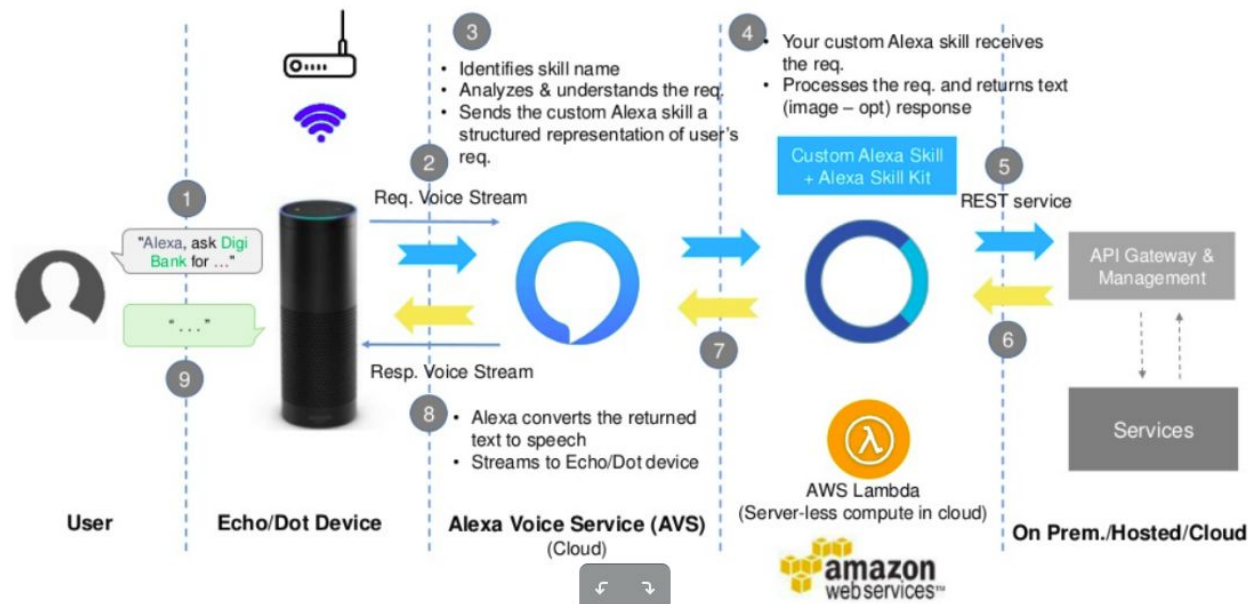


Pre Hackathon Problem: Understand an existing Alexa chatbot	1
Basic definitions:	2
Description of the problem:	4
Who is Asok?	5
Follow all of the following steps in sequence:	5
Setting up Your Alexa Skill in the Developer Console	5
Setting Up A Lambda Function Using Amazon Web Services	6
Connecting Your Voice User Interface To Your Lambda Function	12
Testing Your Alexa Skill	12
Customize the Skill to be Yours	17
PreHackathon-Problem1: Answer the following to your mentor for (3 Marks)	18

Pre Hackathon Problem: Understand an existing Alexa chatbot

Alexa Custom Skill - Reference Architecture



Basic definitions:

For a general idea about chatbots check the following link:

<https://towardsdatascience.com/architecture-overview-of-a-conversational-ai-chat-bot-4ef3dfefd52e>

- A set of *intents* that represent actions that users can do with your skill. These intents represent the core functionality for your skill.
- A set of *sample utterances* that specify the words and phrases users can say to invoke those intents. You map these utterances to your intents. This mapping forms the *interaction model* for the skill.
- An *invocation name* that identifies the skill. The user includes this name when initiating a conversation with your skill.
- Once you have written a few utterances, note the words or phrases that represent variable information. These will become the intent's *slots*. For

example, in the utterances identified earlier, the variables are highlighted in red.

```
i am going on a trip on friday
i want to visit portland
i want to travel from seattle to portland next friday
i'm driving from seattle to portland
i'm driving to portland to go hiking
```

Example conversations:

Before starting conversation lets colour code each of the components of a Chatbot skill:

1. Invocation
2. Utterance
3. Slot value
4. Slot type
5. Intent
6. Response

Skill to Order a book: (Note: All the text in brackets below are explanations about the actual conversation and doesn't involve the conversation itself)

You: Order book

Alexa: Which book do you want to order?

You: I want to order a book by Tolstoy. (Alternate Utterance: Need a book by Tolstoy ⇔ BookOrder)

Alexa: Okay. What is the title of the book?

You: The title of the book is "War and Peace" (Alternate Utterance: Hmm would like to pick War and Peace ⇔ AskBookName)

For the above conversation, the 'Slot Type's are

- 1) **AuthorNames: Tolstoy** you see above is a part of these AuthorNames slot type. The slot can have more values such as **Chekhov, Mark Twain, Amitav Ghosh** etc.
- 2) **BookNames War and Peace** you see above is a part of these BookNames slot type. Can contain other names such as **"Midnight's Children", "Just So Stories"** and so on....

Skill to book a doctor appointment:

You: **Book Doctor Appointment**

Alexa: What the Doctor name and time of the appointment?

You: **I want to book an appointment for 4pm**(Alternate Utterance: **Urgently..for 4 pm 4pm** ⇔ **BookAppointment**)

Alexa: Okay. Which Doctor would you like to consult?

You: **Hmm Dr Adrian Smith preferably.**(Alternate Utterance: **I'll consult Dr Adrian** ⇔ **AsDoctorName**)

Alexa: Okay. Booking confirmed with Dr Adrian Smith at 4pm.

Slot Names and Values:

Time: 1 pm,2pm,3pm,4pm

DoctorNames: Dr Adrian Simith, Dr Peters, Dr Rakesh Khanna

Description of the problem:

In this stage, you will implement an end-to-end Alexa Skill. By the end of this implementation, you should be able to understand the following:

- a. What is a 'Alexa skill'?
- b. What are intents, slots, entities?
- c. What is a lambda function? (In Python)
- d. Finally how end-to-end Alexa Skill works and test it.

The example given here is "PetMatch". Here based on your description of the kind of attributes you look forward to in a Pet, you get recommendations from the chatbot. **You need to implement the following steps** in sequence. This will give you an end-to-end experience of creating an Alexa Skill with Python.

NOTE: It is very important to understand that after you execute the above, you will get an idea about how to check the logs for checking Lamba function (you'll understand about Lambda after you run the above) errors.

Who is Asok?

Asok is just like any one of us [https://en.wikipedia.org/wiki/Asok_\(Dilbert\)](https://en.wikipedia.org/wiki/Asok_(Dilbert)), and his name is used here a few times, if only, for a **dummy name**. Note Asok is case sensitive wherever used.

Follow all of the following steps in sequence:

1. Setting up Your Alexa Skill in the Developer Console

1. Go to the [Alexa Developer Console](#). In the top-right corner of the screen, click the "Sign In" button. (If you don't already have an account, you will be able to create a new one for free.)
2. Once you have signed in, select the Developer Console link and then Alexa Skills Kit.
3. From the Alexa Developer Console select the Create Skill button near the top-right of the list of your Alexa Skills.
4. Give your new skill a Name, for example, 'Pet Match'. This is the name that will be shown in the Alexa Skills Store, and the name your users will refer to.
5. Select the Default Language. This tutorial will presume you have selected 'English (US)'.
6. Select the Custom model under the '*Choose a model to add to your skill*' section. Click the Create Skill button at the top right.
7. Choose Start from scratch from the *Choose a template* section and click the Choose button on the top right.
8. Build the Interaction Model for your skill
 - On the left hand navigation panel, select the JSON Editor tab under Interaction Model. In the textfield provided, replace any existing code with the code provided in the [Interaction Model](#). Click Save Model.
 - If you want to change the skill invocation name, select the Invocation tab. Enter a Skill Invocation Name. This is the name that your users will need to say to start your skill. In this case, it's preconfigured to be 'pet match'.
 - Click "Build Model".
9. Note: You should notice that Intents and Slot Types will auto populate based on the JSON Interaction Model that you have now applied to your skill. Feel free to

explore the changes here, to learn about Intents, Slots, and Utterances open our [technical documentation in a new tab](#).

10. Optional: Select an intent by expanding the Intents from the left side navigation panel. Add some more sample utterances for your newly generated intents. Think of all the different ways that a user could request to make a specific intent happen. A few examples are provided. Be sure to click Save Model and Build Model after you're done making changes here.
11. If your interaction model builds successfully, proceed to the next step. If not, you should see an error. Try to resolve the errors. In our next step of this guide, we will be creating our Lambda function in the AWS developer console, but keep this browser tab open, because we will be returning here on [Page #3: Connect VUI to Code](#).
12. If you get an error from your interaction model, check through this list:
 - Did you copy & paste the provided code correctly?
 - Did you accidentally add any characters to the Interaction Model?

2. Setting Up A Lambda Function Using Amazon Web Services

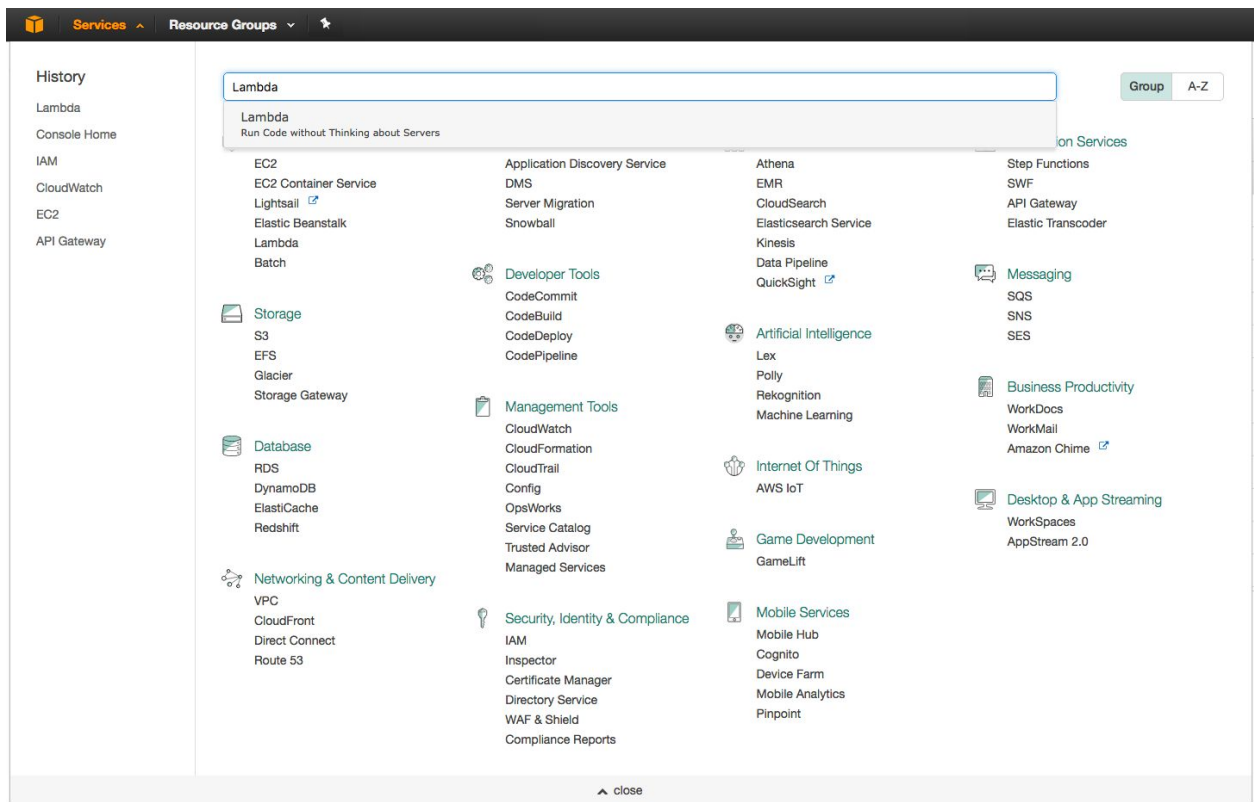
In the [first step of this guide](#), we built the Voice User Interface (VUI) for our Alexa skill. On this page, we will be creating a Lambda function using [Amazon Web Services](#). You can [read more about what a Lambda function is](#), but for the purposes of this guide, what you need to know is that Lambda is where our code lives. When a user asks Alexa to use our skill, it is our Lambda function that interprets the appropriate interaction, and provides the conversation back to the user.

1. CRITICAL STEPS:

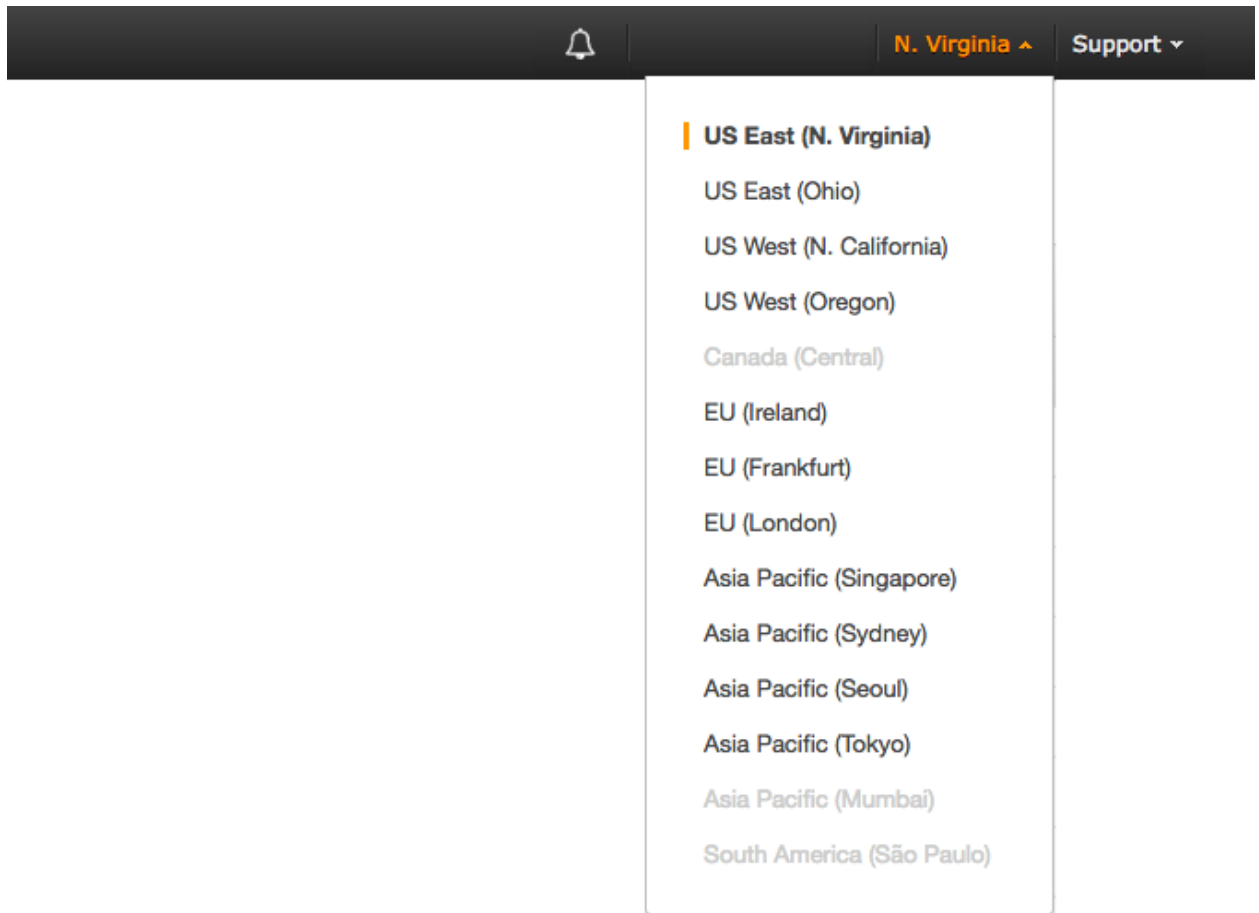
- i. Login to `aiml-sandbox1.talentsprint.com` using your putty/filezilla username passwords (i.e. `b7h2<teamNbr>`)
 - ii. Run “ **source venv/bin/activate** “ to activate your Python 2.7 virtual environment. This step is “extremely critical”
 - iii. Now change directory (`cd`) to “Hackathon” folder. You will need to access this folder again a few steps later.
2. Go to <http://aws.amazon.com> and sign in to the console. If you don't already have an account, you will need to create one. [Check out this quick walkthrough for setting up a new AWS account](#).

Sign In to the Console

3. Choose **"Services"** at the top of the screen, and type "Lambda" in the search box. You can also find it in the list of services. It is in the "Compute" section.



4. Check your **AWS region**. Lambda only works with the Alexa Skills Kit in four regions: **US East (N. Virginia)**, EU (Ireland), US West (Oregon) and Asia Pacific (Tokyo). Make sure you choose the region closest to your customers (you will choose this on the top right of **AWS management console** i.e. home page of aws.amazon.com after signing in). You are advised to use **US East (N. Virginia)**



5. Now configure your aws access with the following commands (**NOTE: FOR THIS YOU NEED TO GO BACK THE AIML-SANDBOX1.TALENTSPRINT.COM that you setup in “[CRITICAL STEPS](#)” and run the commands below**):
- i. Run the command “ **aws configure --profile AsokXYZ** ” (each of your team members can configure their own profile if needed. However, if you have created a common account for your team, configuring one profile is enough. In either case replace AsokXYZ with your own profile name). You’ll have to fill in the following details: (keep them ready beforehand. “Default region name” can be got from the URL of your aws.amazon.com after login. For example: (see **us-east-1** in bold in this URL. That is the region name (<https://console.aws.amazon.com/iam/home?region=us-east-1#/home>)
 - i. **AWS Access Key ID [None]:**
 - ii. **AWS Secret Access Key [None]:**
 - iii. **Default region name [None]:**

iv. Default output format [None]: **set this value to “json” in lower case.**

ii. **Then do remember to export it with the following:**

```
export AWS_PROFILE=AsokXYZ
```

iii. After performing the above steps you will be able to see your profile under “**vi ~/.aws/credentials**”

6. **Click the "Create a Lambda function" button.** It should be near the top of your screen.



7. **Click on "Author from scratch".** We will configure our Lambda function next.

i. These values will only ever be visible to you, but make sure that you name your function something meaningful. "samplePythonPetMatch" is sufficient if you don't have another idea for a name.

ii. From the "Runtime" dropdown select the python version your system supports. This tutorial and sample code works with either Python 2.7 or 3.6. To check the python version, try the following command in a terminal

iii. `$ python --version`

```
Python 2.7.10
```

iv. Set up your Lambda function role. If you haven't done this before, we have a [detailed walkthrough for setting up your first role for Lambda](#). If you have done this before, you only need to select the Existing role.

v. Click Create function.

8. **Configure your trigger.** There are many different AWS services that can trigger a Lambda function, but for the purposes of this guide, we need to select "Alexa Skills Kit." from the left hand side.

Once you have selected Alexa Skills Kit, scroll down and find the Skill ID verification section. Although you will want to paste your skill's ID in the Skill ID

field, however for this tutorial, click Disable. Click the Add button in the lower right. Click the orange Save button in the top right corner.

9. **Finish configuring your function.** Click on your function's name (you'll find it in the middle) and scroll to the bottom of the page, you'll see a Cloud9 code editor.

We have provided the code for this skill [here](#). To properly upload this code to Lambda, you'll need to perform the following:

- i. Go back to “Hackathon” folder that you were ready with at end of **“CRITICAL STEPS”**
- ii. This skill uses the [ASK SDK for Python](#) for development. The skill code is provided in the [lambda_function.py](#), and the dependencies are mentioned in [requirements.txt](#). Download the two files discussed above, by running the following wget commands:
 - i. `wget https://raw.githubusercontent.com/alexa/skill-sample-python-petmarch/master/lambda/py/requirements.txt`
 - ii. `wget https://raw.githubusercontent.com/alexa/skill-sample-python-petmarch/master/lambda/py/lambda_function.py`
- iii. On your system, navigate to the lambda folder and install the dependencies in a new folder called “skill_env” using the following command:
- iv.

```
pip install -r requirements.txt -t skill_env
```
- v. Copy the lambda_function.py into the skill_env folder.
- vi.

```
cp lambda_function.py ./skill_env/
```
- vii. Zip the contents of the skill_env folder. **Remember to zip the contents of the folder and NOT the folder itself.**
 - i. For this run the following command inside skill_env folder (i.e. first cd to skill_env folder) and then run “**zip -r asokZip.zip ***”
- viii. Now create a AWS bucket and upload the zip file to the bucket. Instruction below (this is so that we can access this uploaded file in Lambda):

- i. Create the bucket:

Run the following command on the
aiml-sandbox1.talentsprint.com **aws s3 mb
s3://asok-bucket --region=us-east-1** (replace asok-bucket,
with your own bucket name. Also, the region has to be one
of the optional regions mentioned [above](#), preferably
us-east-1 Note: Bucket names can contain **lowercase
letters**, numbers, hyphens, and periods. Bucket names can
start and end only with a letter or number, and cannot
contain a period next to a hyphen or another period.).

- ii. Copy the zip file to bucket: Here is the command. Run it at the
location where you have the zip file.

```
aws s3 cp filename.zip s3://your-bucket-name/ --grants  
read=uri=http://acs.amazonaws.com/groups/global/AllUsers  
full=emailaddress=GiveYourMailid
```

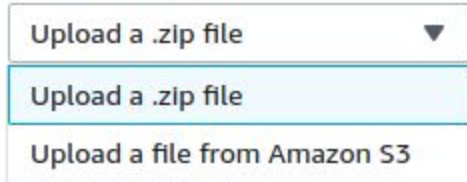
- iii. Check whether the upload is successful by running the following:

```
aws s3 ls s3://asok-bucket (replace asok-bucket with your  
own bucket name)
```

(Optional) Follow the ASK Python SDK [Getting Started](#) documentation, to check
alternative ways of installing the sdk and deploying to AWS Lambda console.

10. Now go to your Lambda function and select “**Upload a file from Amazon S3**”

Code entry type



Upload a .zip file ▼
Upload a .zip file
Upload a file from Amazon S3

And then give the link to the bucket such as

<https://s3.amazonaws.com/asok-bucket/path/to/asokLambda.zip>

11. (Optional) **Click the Configure test events** dropdown menu on the top of the page.
12. Select 'Alexa Start Session' from the 'Event Template' dropdown.
13. Type `LaunchRequest` into the 'Event Name' field.
14. **Click the orange 'Create' button** at the bottom of the page
15. **Click the Test button** at the top of the page. 1. You should see a light green box with the message: *Execution result: succeeded* at the top of the page.
16. **VERY IMP: As a final step, copy the ARN value from the top right corner of the screen.** You will need this value in the next section of this guide.

3. Connecting Your Voice User Interface To Your Lambda Function

On [page #1](#) of this guide, we created a voice user interface for the intents and utterances we expect from our users. On [page #2](#), we created a Lambda function that contains all of our logic for the skill. On this page, we need to connect those two pieces together.

1. Go back to the [Amazon Developer Portal](#) and select your skill from the list. You may still have a browser tab open if you started at the beginning of this tutorial.

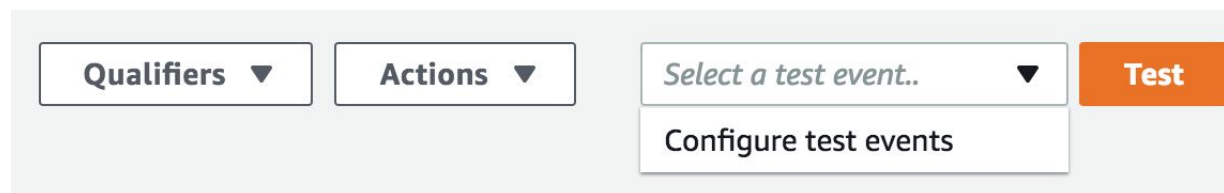
2. Select the Endpoint tab on the left side navigation panel.
3. Select the "AWS Lambda ARN" option for your endpoint. You have the ability to host your code anywhere that you would like, but for the purposes of simplicity and frugality, we are using AWS Lambda. ([Read more about Hosting Your Own Custom Skill Web Service](#).) With the AWS Free Tier, you get 1,000,000 free requests per month, up to 3.2 million seconds of compute time per month. Learn more at <https://aws.amazon.com/free/>. In addition, Amazon now offers [AWS Promotional Credits for developers who have live Alexa skills that incur costs on AWS related to those skills](#).
4. Paste your Lambda's ARN (Amazon Resource Name) into the textbox provided for Default Region.
5. **Click the Save Endpoints button** at the top of the main panel.
6. **Click the "Next" button** to continue to the next section below.

4. Testing Your Alexa Skill

So far, we have [created a Voice User Interface](#) and [a Lambda function](#), and [connected the two together](#). Your skill is now ready to test.

1. Go back to the [Amazon Developer Portal](#) and select your skill from the list. You may still have a browser tab open if you started at the beginning of this tutorial.
2. Open the Test Pane, by selecting the Test link from the top navigation menu.
3. Enable Testing by activating the Test is enabled for this skill slider. It should be underneath the top navigation menu.
4. To validate that your skill is working as expected, invoke your skill from the Alexa Simulator. You can either type or click and hold the mic from the input box to use your voice.
 - Type "Open" followed by the invocation name you gave your skill in [Step 1](#). For example, "Open Pet Match".
 - Use your voice by clicking and holding the mic on the side panel and saying "Open" followed by the invocation name you gave your skill.
 - If you've forgotten the invocation name for your skill, revisit the Build panel on the top navigation menu and select Invocation from the sidebar to review it.
5. Ensure your skill works the way that you designed it to.

- After you interact with the Alexa Simulator, you should see the Skill I/O JSON Input and JSON Output boxes get populated with JSON data. You can also view the Device Log to trace your steps.
 - If it's not working as expected, you can dig into the JSON to see exactly what Alexa is sending and receiving from the endpoint. If something is broken, AWS Lambda offers an additional testing tool to help you troubleshoot your skill.
6. Configure a test event in AWS Lambda. Now that you are familiar with the request and response boxes in the Service Simulator, it's important for you to know that you can use your requests to directly test your Lambda function every time you update it. To do this:
- Enter an utterance in the service simulator, and copy the generated Lambda Request for the next step.
 - Open your Lambda function in AWS, open the Actions menu, and select "Configure test events."



-
- Select "Create New Test Event". Choose "Alexa Start Session" as the Event Template from the dropdown list. You can choose any test event in the list, as they are just templated event requests, but using "Alexa Start Session" is an easy one to remember.

Configure test event



A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

- ☒ Create new test event
☐ Edit saved test events

Event template

Alexa Start Session ▼

Q |

Step Functions Error

CloudFront Response Generation

Kinesis Firehose Apache Log

S3 Delete

Alexa

Alexa Smart Home - Discovery

Alexa Intent - GetNewFact

Alexa Start Session

Alexa Intent - Recipe

Alexa Smart Home - Turn Off

```
14  "request": {  
15    "locale": "en-US",  
16    "timestamp": "2016-10-27T18:21:44Z",  
17    "type": "LaunchRequest",  
18    "requestId": "amzn1.echo-api.request.[unique-value-here]"  
19  },  
20  "context": {  
21    "AudioPlayer": {  
22      "playerActivity": "IDLE"  
23    },  
24    "System": {  
25      "device": {  
26        "supportedInterfaces": {  
27
```

Cancel

Create

-
- Type in an Event Name into the Event Name Dialog box. Delete the contents of the code editor, and paste the Lambda request you copied above into the code editor. The Event Name is only visible to you. Name your test event something descriptive and memorable. For our example, we entered an event name as "startSession". Additionally, by copying and pasting your Lambda Request from the service simulator, you can test

different utterances and skill events beyond the pre-populated templates in Lambda.

Input test event



Use the editor below to enter an event to test your function with. You can edit the event again by choosing **Configure test event** in the Actions list. Note that changes to the event will only be saved locally.

Sample event template

Alexa Start Session



```
1 {
2   "session": {
3     "sessionId": "SessionId.ce5b9874-0f86-49d0-8fe2-85f5a2008386",
4     "application": {
5       "applicationId": "amzn1.ask.skill.ac4240ad-4b18-484a-ab67-d740ed1320ac"
6     },
7     "attributes": {},
8     "user": {
9       "userId": "amzn1.ask.account.AGZFAKNV3GFD50WVXLULBF2NNRHUSJEHVDEMNQ2ZHTN5N6FPC"
10    },
11    "new": true
12  },
13  "request": {
14    "type": "IntentRequest",
15    "requestId": "EdwRequestId.1181fc5e-453e-4bb4-8ec5-ba48bcdf88ec",
16    "locale": "en-US",
17    "timestamp": "2017-05-03T15:09:42Z",
18    "intent": {
19      "name": "GetNewFactIntent",
20      "slots": {}
21    }
22  },
23  "version": "1.0"
24 }
```

Cancel

Save

Save and test

-
- Click the "Create" button. This will save your test event and bring you back to the main configuration for your lambda function.
- Click the "Test" button to execute the "startSession" test event.

Actions ▼

startSession ▼

Test

-
- This gives you visibility into four things:
 - Your response, listed in the "Execution Result."

Execution result: succeeded (logs)

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
{
  "version": "1.0",
  "response": {
    "outputSpeech": {
      "type": "SSML",
      "text": "Speak: Here's your fact: Despite being further from the Sun, Venus experiences higher temperatures than Mercury. </speak>"
    },
    "shouldEndSession": true,
    "name": {
      "type": "Simple"
    }
  }
}
```

- A Summary of the statistics for your request. This includes things like duration, resources, and memory used.

Summary

Code SHA-256 0zWc5zYBiftT7c54Ugb/sg/p+PjBLF/re8q2wnpr2M=

Request ID ebe2b808-fd25-11e6-b77c-d1c7c6d5916c

Duration 72.45 ms

Billed duration 100 ms

Resources configured 128 MB

Max memory used 21 MB

- Log output. By effectively using print() or python logger statements in your Lambda code, you can track what is happening inside your function, and help to figure out what is happening when something goes wrong. You will find the log to be incredibly valuable as you move into more advanced skills.

Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda. [Click here](#) to view the CloudWatch log group.

```
START RequestId: ebe2b808-fd25-11e6-b77c-d1c7c6d5916c Version: $LATEST
2017-02-27T19:49:59.885Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      Warning: Application ID is not set
2017-02-27T19:49:59.887Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----LAUNCH REQUEST.  SETTING STATE == START.  REDIRE
2017-02-27T19:49:59.943Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----ANSWER INTENT.
2017-02-27T19:49:59.943Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----{"session":{"sessionId":"SessionId.457b2d2e-5014
2017-02-27T19:49:59.943Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----StateName,Abbreviation,Capitol,StatehoodYear,Sta
2017-02-27T19:49:59.943Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----CHECKING SLOT: Abbreviation
2017-02-27T19:49:59.943Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----CHECKING SLOT: StatehoodYear
2017-02-27T19:49:59.943Z      ebe2b808-fd25-11e6-b77c-d1c7c6d5916c      -----CHECKING SLOT: StateName
```

- A link to your [CloudWatch](#) logs for this function. This will show you all of the responses and log statements from every user interaction. This is very useful, especially when you are testing your skill from a device with your voice. (It is the "[Click here](#)" link in the Log Output description.)

7. Other testing methods to consider:

- [Echosim.io](#) - a browser-based Alexa skill testing tool that makes it easy to test your skills without carrying a physical device everywhere you go.

- [Unit Testing with Alexa](#) - a modern approach to unit testing your Alexa skills with [Postman](#) and [Amazon API Gateway](#).
8. If your sample skill is working properly, you can now customize your skill.

5. Customize the Skill to be Yours

At this point, you should have a working copy of our Pet Match skill. In order to make it your own, you will need to customize it with data and responses that you create. Here are the things you will need to change:

1. New sentences to respond to your users. There are several sentences and responses that you will want to customize for your skill.
 - Open a copy of `lambda_function.py`. If you haven't already downloaded the code for this project, [you can find a copy of lambda_function.py here](#). You can use a simple, lightweight code editor like [Atom](#), [Sublime Text](#), or [VSCode](#). Or Vi.
 - Look for lines like this: `speech = ('Welcome to pet match...'` These are strings that hold phrases for Alexa to respond with. Customize them to make it as varied and conversational as time allows.
 - Continue through `lambda_function.py` until you reach the bottom of the file. This will ensure that you cover each of the Alexa responses that you need to update.
 - When you have replaced the data in `lambda_function.py`, you need to upload the latest data into Lambda. Copy the updated contents into the `skill_env` folder, zip the contents of the `skill_env` folder and upload it to AWS Lambda as discussed in the "Finish configuring your function" step in [Lambda setup documentation](#). Test your skill through the Alexa Simulator on the developer portal, with the updated changes.
2. New API. The sample skill uses a custom API to get pet matches. Update it with your API, for better customization and handling.
 - Go back to your copy of [lambda_function.py](#).
 - Look for `pet_match_api` This is the API used to get pet matches. Change it according to your API call.
 - When you have replaced the api details in `lambda_function.py`, you need to upload the latest data into Lambda. Copy the updated contents into the

`skill_env` folder, zip the contents of the `skill_env` folder and upload it to AWS Lambda as discussed in the "Finish configuring your function" step in [Lambda setup documentation](#). Test your skill through the Alexa Simulator on the developer portal, with the updated changes.

3. New language. If you are creating this skill for another language other than English, you will need to make sure Alexa's responses are also in that language.
 - For example, if you are creating your skill in German, every single response that Alexa makes has to be in German. You can't use English responses or your skill will fail certification.

PreHackathon-Problem1: Answer the following to your mentor for **(3 Marks)**

- What is an invocation call? Create 3 possible ones?
- For a given Intent, can the Intent names be non-unique? I.e. Can a particular intent be called by different 'Intent names'?
- Are multiple Utterances possible for a given intent?
- What is the difference between a "Slot Value" and a "Slot Name"?
- Lambda functions are nothing but the JSON output explaining the conversation flow? TRUE/FALSE
- Experiment with changes in Intent, Slot, Dialog flow and see how it changes the JSON file in the **JSON Editor** tab