

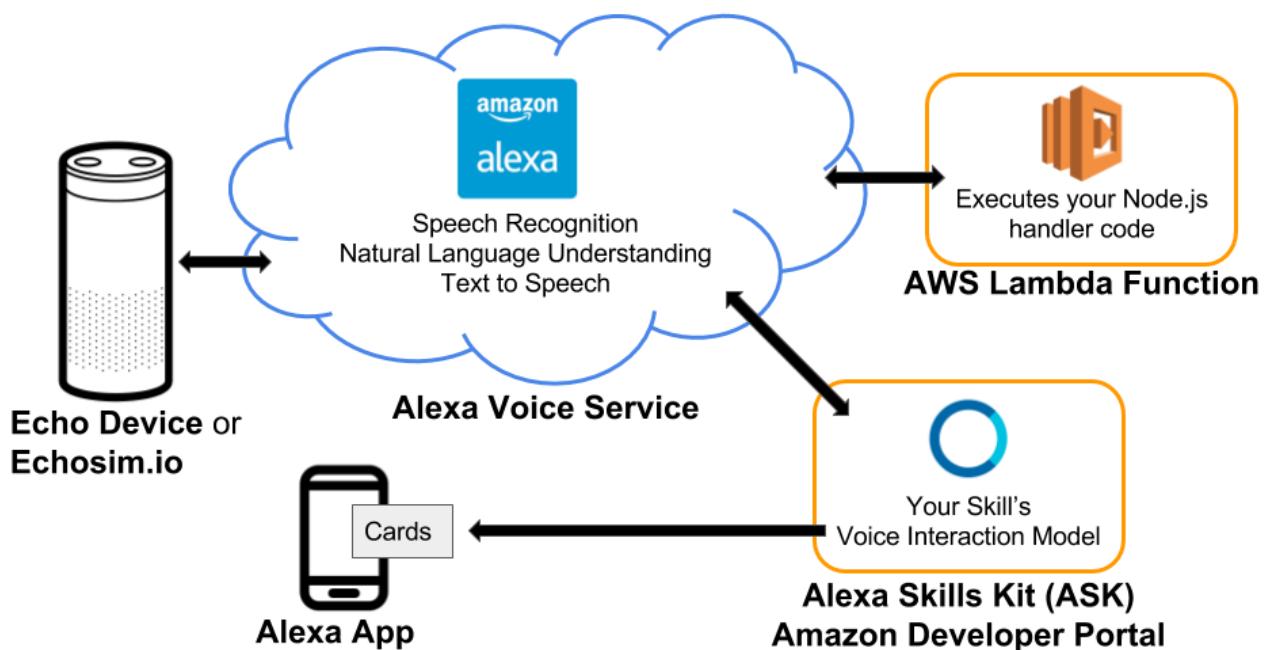


Overview

It's time to create your own Alexa Skill!. In this project, you'll build a fully functional skill for [Amazon's Alexa](#) that provides year-dated facts from AI History (or some other History of your choosing). Through the process, you'll get to use the [Alexa Skills Kit \(ASK\)](#) - a current state of the art API for building voice systems.

Some basic template code similar to the Space Geek lab has been provided for you. Your skill for this project will be a history skill, however, that also allows users to ask for facts from a particular year. You are free to choose any category you wish for your facts. For example, you could have a skill that provides historical facts about a sport, or a location, or hobby, but each fact you provide must include a four-digit year.

If you are not sure what category you want to try, don't worry...an example file has been provided with a few historical facts about artificial intelligence to get you started. It can be hard to come up with a robust list of suitable facts - feel free to crowd source a list of historical facts with your fellow students!



Getting Started

Set up your Amazon accounts

We highly recommend you complete the [Space Geek lab](#), which steps you through setting up your Amazon Developer and Amazon AWS accounts as well as building the project this one is based on.

[Preview] Project: Alexa History Skill

one already.

- Navigate to [Amazon AWS](#) and create an account if you do not already have one. You will need to provide a credit card to set up the account, but new accounts receive the [AWS Free Tier](#) which should suffice for this project as it includes one million non-expiring AWS Lambda requests per month. Note: Lambda functions for Alexa skills can be hosted in either the US East (N. Virginia) or EU (Ireland) region. These are the only regions the Alexa Skills Kit supports, so you may need to change this setting in the upper right portion of your screen on the console.

Starter Code

Install your starter code locally.

- Download or clone the [starter code](#) from GitHub
- Save it in a directory named **AIND-VUI-Alexa**. It contains the following directories and files:
 - **speechAssets/IntentSchema.json** - intents definition for the interactive model
 - **speechAssets/SampleUtterances_en_US.txt** - utterances for the interactive model
 - **src/index.js** - skill logic and handlers to be run in AWS Lambda
 - **src/facts.js** - a list of facts that the skill will use in responses
 - **tests/*.js** - various unit tests to be run locally with mocha; you do not need to change these

Environment

1. Install [Node.js](#) per instructions on the website for your machine.

2. Install dependencies for the project

- Navigate to the **AIND-VUI-Alexa/src** directory of the starter code and open a terminal window.
- The dependencies we need are listed in the **package.json** file and include the [alexa-sdk](#) library for Alexa as well as [mocha](#) test framework for Node.js along with [chai](#) and [aws-lambda-mock-context](#) for local unit testing. Install them all with the following command:

```
$ npm install
```

- There should now be a directory named **node_modules** within the **src** directory. This is how Node.js attaches libraries for your code.

3. Unit testing

- You can now run the provided unit tests from the command line within the **src** directory with the following command. Try it now:

```
$ npm test
```

[Preview] Project: Alexa History Skill

tests should pass and all others should fail. You may have to scroll up to see the passing tests. As you complete the tasks ahead, you can use these unit tests (and write more yourself) to quickly test code changes prior to deployment to AWS Lambda.

4. JavaScript coding

- At this point, you can open your project with a code editor of your choice. Some free ones that support JavaScript and Node.js include [Atom](#) and [Visual Studio Code](#)
- Udacity has a free [Intro to JavaScript](#) course available that will quickly bring you up to speed on JS syntax. The JavaScript skills needed for this project primarily require following patterns you find in the existing starter code and the use of general coding constructs such as for loops, conditional statements, and arrays. Here are a few links that might be helpful for quick reference:
 - [js for loops](#)
 - [js conditional statements](#)
 - [js arrays](#)
 - [js String includes\(\) method](#)
 - [js JSON](#)
 - [Alexa skill examples](#)
 - [Node.js API reference](#)

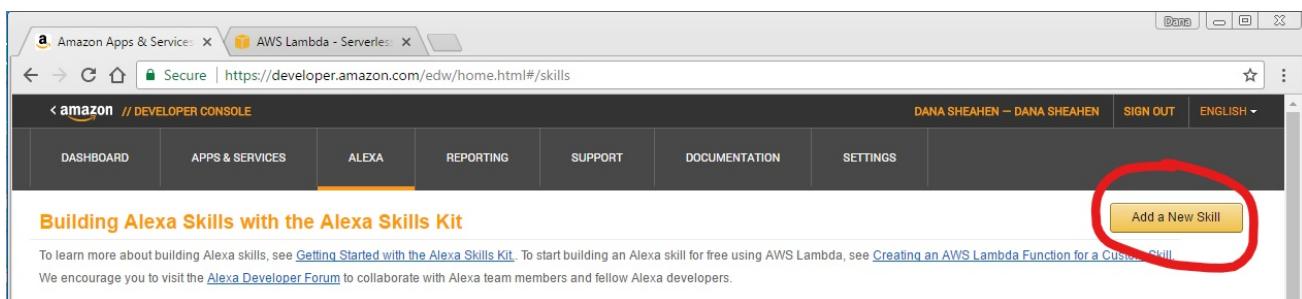
5. Run the Starter Code on AWS Lambda

- Deploy the starter code to verify that it works with your accounts in its simple form. This is the same process you went through with the Space Geek Lab. If you need a refresher, step-by-step instructions are provided [here](#).

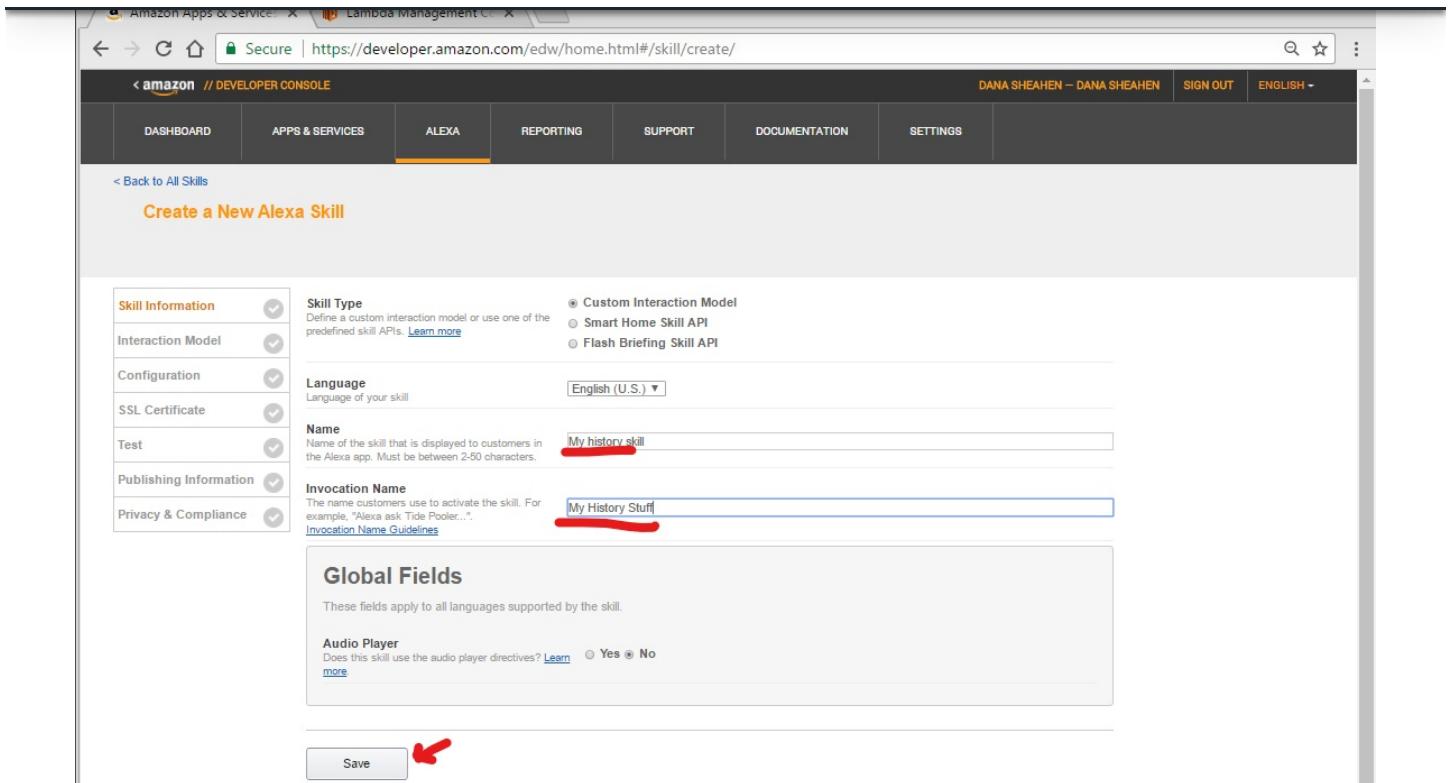
Deploying your Skill

The Space Geek Lab steps you through setting up an Alexa Skill. The starter code can be deployed in the same way. The following step-by-step instructions assume you already have completed the steps in [Getting Started](#), and are ready to deploy your starter code.

1. Open both the Amazon Developer portal and the Amazon AWS console
2. On the Developer portal, navigate to Create the new skill with the Alexa Skills Kit in the Amazon Developer portal, and "Add a New Skill":



[Preview] Project: Alexa History Skill



Secure | https://developer.amazon.com/edw/home.html#/skill/create/

DASHBOARD APPS & SERVICES ALEXA REPORTING SUPPORT DOCUMENTATION SETTINGS

< Back to All Skills

Create a New Alexa Skill

Skill Information

Skill Type
Define a custom interaction model or use one of the predefined skill APIs. [Learn more](#)

Custom Interaction Model
 Smart Home Skill API
 Flash Briefing Skill API

Interaction Model

Configuration

SSL Certificate

Test

Publishing Information

Privacy & Compliance

Language
Language of your skill [English \(U.S.\) ▾](#)

Name
Name of the skill that is displayed to customers in the Alexa app. Must be between 2-50 characters.
My history skill

Invocation Name
The name customers use to activate the skill. For example, "Alexa ask Tide Pooler..." [Invocation Name Guidelines](#)

Global Fields

These fields apply to all languages supported by the skill.

Audio Player
Does this skill use the audio player directives? [Learn more](#) Yes No

Save 

4. In the Interaction Model, copy/paste the contents of the **IntentSchema.json** file into the "Intent Schema" text box and the contents of the **SampleUtterances_en_US.txt** file into the "Sample Utterances" text box. Then "Save" it and click "Next".

[Preview] Project: Alexa History Skill

The screenshot shows the configuration page for the Alexa History Skill. On the left, there is a sidebar with sections: Interaction Model (checked), Configuration (checked), Test (checked), Publishing Information (checked), and Privacy & Compliance (checked). Below this is a box for Skills Beta Testing (NEW) with the status: Not yet eligible. The main content area has a heading "Intent Schema" with a description: "The schema of user intents in JSON format. For more information, see [Intent Schema](#). Also see [built-in slots](#) and [built-in intents](#)". A "Launch Skill Builder BETA" button is present. A large red arrow points to the JSON code in the Intent Schema section:

```

1 {
2   "intents": [
3     {
4       "intent": "GetNewFactIntent"
5     },
6     {
7       "intent": "AMAZON.HelpIntent"
8     },
9     {
10      "intent": "AMAZON.StopIntent"
11    },

```

Below this is a "Custom Slot Types (Optional)" section with a "Enter Type" input field and a "TYPE" placeholder. A "Enter Values" section follows, with a note: "Values must be line-separated". A large red arrow points to the "Enter Values" input field:

```

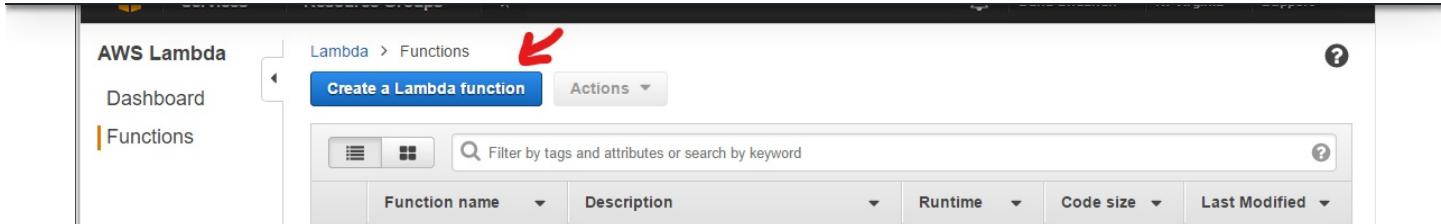
1

```

At the bottom of the page are buttons: "Cancel", "Add", "Save" (disabled), "Submit for Certification" (disabled), and a large yellow "Next" button. A red arrow points to the "Next" button.

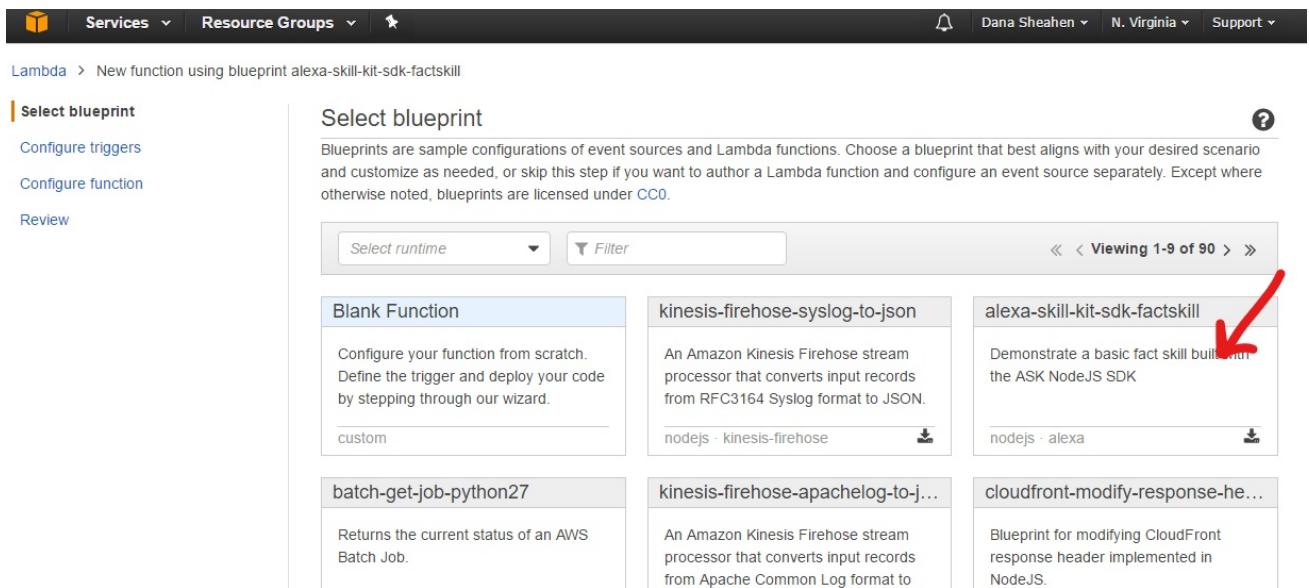
5. From the src directory of your starter code, zip index.js, facts.js, and the node_modules directory together. This zip file will be uploaded next to AWS Lambda.
6. Navigate to AWS console and choose the AWS Lambda Service. Click "Create a Lambda Function".

[Preview] Project: Alexa History Skill



The screenshot shows the AWS Lambda Functions page. The left sidebar has 'AWS Lambda' selected. The main area shows a table with columns: Function name, Description, Runtime, Code size, and Last Modified. A red arrow points to the 'Create a Lambda function' button at the top of the page.

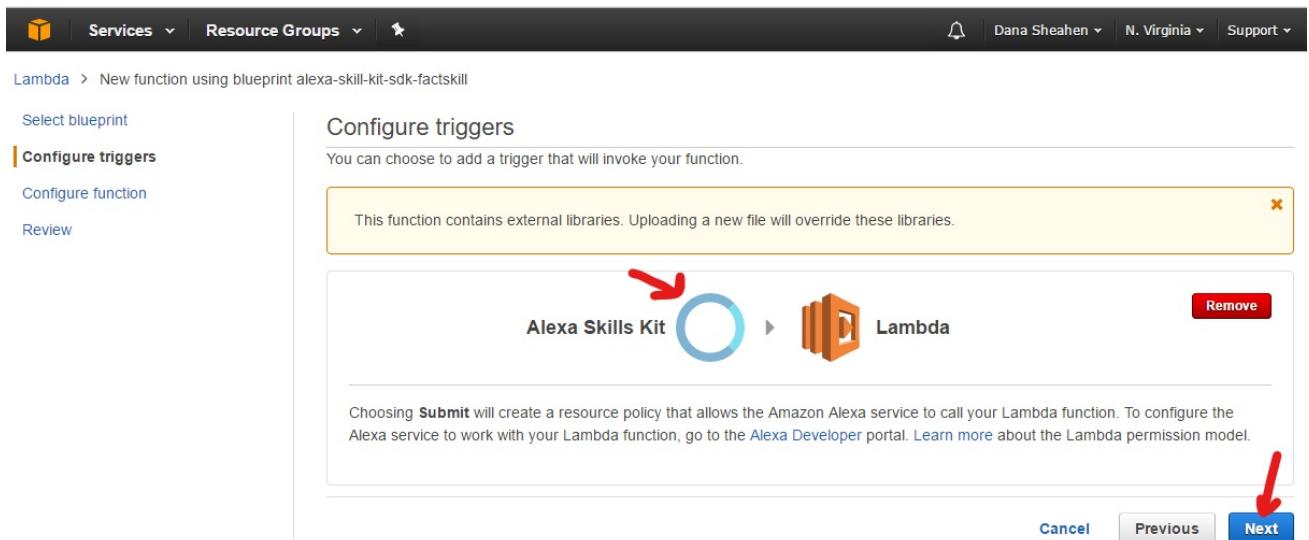
7. Select the "alexa-skill-kit-sdk-factskill" blueprint.



The screenshot shows the 'Select blueprint' step in the Lambda wizard. The left sidebar shows 'Select blueprint' selected. The main area displays a list of blueprints. A red arrow points to the 'alexa-skill-kit-sdk-factskill' card, which is described as 'Demonstrate a basic fact skill built with the ASK NodeJS SDK'.

Blueprint	Description
Blank Function	Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard.
kinesis-firehose-syslog-to-json	An Amazon Kinesis Firehose stream processor that converts input records from RFC3164 Syslog format to JSON.
alexa-skill-kit-sdk-factskill	Demonstrate a basic fact skill built with the ASK NodeJS SDK
batch-get-job-python27	Returns the current status of an AWS Batch Job.
kinesis-firehose-apachelog-to-j...	An Amazon Kinesis Firehose stream processor that converts input records from Apache Common Log format to
cloudfront-modify-response-he...	Blueprint for modifying CloudFront response header implemented in NodeJS.

8. Click the gray box to configure the trigger (the skill we defined in the Developer portal) and choose "Alexa Skills Kit" from the drop down menu. Click "Next".



The screenshot shows the 'Configure triggers' step in the Lambda wizard. The left sidebar shows 'Configure triggers' selected. The main area shows a 'Configure triggers' section with a note: 'You can choose to add a trigger that will invoke your function.' Below is a list of triggers. A red arrow points to the 'Alexa Skills Kit' trigger icon. Another red arrow points to the 'Next' button at the bottom right.

Trigger
Alexa Skills Kit
Lambda

Choosing **Submit** will create a resource policy that allows the Amazon Alexa service to call your Lambda function. To configure the Alexa service to work with your Lambda function, go to the Alexa Developer portal. [Learn more about the Lambda permission model.](#)

9. On the Configure Function page, give your function a name and description.

[Preview] Project: Alexa History Skill

Configure function

A Lambda function consists of the custom code you want to execute. Learn more about Lambda functions.

Name* ←

Description ←

Runtime* ▼

10. Scrolling down to the “Lambda function handler and role”, choose the existing `lambda_basic_execution` role you set up during the lab. Click “Next”. If you did not do that previously, you will need to review the [Amazon instructions](#) to set one up.

Lambda function handler and role

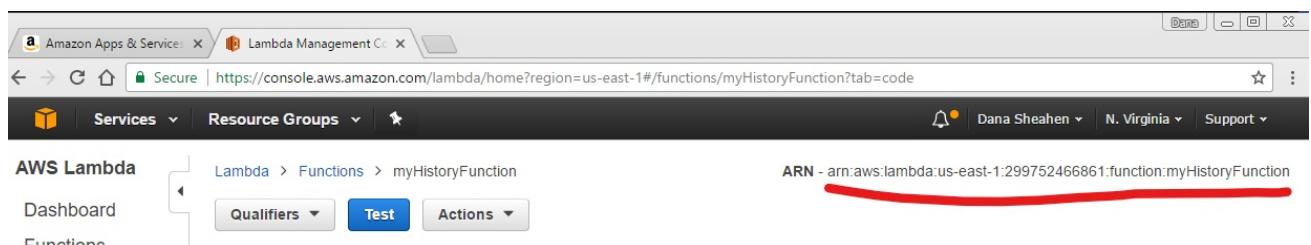
Handler* i

Role* ▼ i

Existing role* ▼ i

11. On the Review page, click “Create function”. You should get a new page that includes a “Congratulations!”.

12. Upload the zip file you created earlier with the starter code files. Do this by selecting the “Code entry type” drop down menu as “Upload a ZIP file”. Click the “Upload” button to find your file and then click “Save”.



The screenshot shows the AWS Lambda Management Console. The URL in the browser is <https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/myHistoryFunction?tab=code>. The page displays the Lambda function details for 'myHistoryFunction'. The ARN of the function is highlighted with a red box: `arn:aws:lambda:us-east-1:299752466861:function:myHistoryFunction`.

[Preview] Project: Alexa History Skill

English (U.S.) Add a New Language

Global Fields
These fields apply to all languages supported by the skill.

Endpoint

Service Endpoint Type: AWS Lambda ARN (Amazon Resource Name) Recommended HTTPS

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: i

North America Europe

North America
arn:aws:lambda:us-east-1:299752466861:function:

14. The starter code is ready to try. Enter “a fact” in the Text portion of the Service Simulator. If everything is set up correctly, you will see both a Lambda Request and a Lambda Response.

Service Simulator

Use Service Simulator to test your lambda function: [arn:aws:lambda:us-east-1:299752466861:function:myHistoryFunction](#) ▾

Note: Service Simulator does not currently support testing audio player directives and customer account linking.

Text JSON

Enter Utterance

a fact

Ask My history skill Reset

Lambda Request

```

1 {
2   "session": {
3     "sessionId": "SessionId.0e18e0cf-ee09-41f3-9c
4     "application": {
5       "applicationId": "amzn1.ask.skill.ebd1f1a1-
6     },
7     "attributes": {},
8     "user": {
9       "userId": "amzn1.ask.account.AFA5K3Y2EE4BFRF
10     },
11     "new": true
12   },
13   "request": {
14     "type": "IntentRequest",
15     "requestId": "EdwRequestId.6b33f388-bbda-4afe-
16

```

Lambda Response

```

1 {
2   "version": "1.0",
3   "response": {
4     "outputSpeech": {
5       "type": "SSML",
6       "ssml": "<speak> Here's your fact: The t
7     },
8     "card": {
9       "content": "The term, Artificial Intelligence, wa
10      "title": "My American History Facts",
11      "type": "Simple"
12    },
13    "shouldEndSession": true
14

```

Listen

15. Throughout the project, you can continue to test your results this way or with an Echo device by uploading a new zip file with your changes. Since this is a bit tedious for every change, you may find it easier to test small changes with the local unit tests.

[Preview] Project: Alexa History Skill

been provided in an external file, **facts.js**, instead of the space facts. Each fact includes a 4-digit year in its text, which we will use in the project for a new feature.

The project consists of three parts:

1. Customize the fact skill
2. Add a feature using an additional intent and a slot
3. Add conversational elements

Part 1: Customize the fact skill

1. Choose a history category you wish to use for your skill. You can continue to use the AI History Facts already started for you if you wish.
2. Expand the utterances in the **speechAssets/SampleUtterances_en_US.txt** file to include at least 15 appropriate utterances for **GetNewFactIntent**. Examples can be found [here](#).
3. Expand the facts list in **facts.js** such that there are at least 10 distinct facts, where each includes a 4-digit year in its text. These will be spoken by the Amazon Text-To-Speech algorithm (TTS), so keep in mind where you wish pauses to occur. To hear how it sounds, enter your sentence in the developer portal under the “Test” section:

English (U.S.) Add a New Language

Skills Information

Interaction Model

Configuration

Test

Publishing Information

Privacy & Compliance

Skills Beta Testing NEW
Status: Not yet eligible ?

Interaction Model

(i) Please complete the Interaction Model tab to start testing this skill.

Enabled This skill is enabled for testing on your account. ?

Once you have completed testing on your device, please complete the Description and Publishing Information tab, then submit the skill for certification.

If it passes Amazon's testing and certification process, it will become available to Alexa end users.

You will be able to see your skill in the Skills tab in Alexa App and you can enable the skill and start testing.

After completing your testing please submit the skill for certification. If it passes Amazon's testing and certification process, it will become available to Alexa end users.

The skill is available in "Skills > Your Skills" page of the Alexa App when you select 'Yes' above. You can then enable the skill and test its functionality by asking Alexa, **ask arty**

Voice Simulator

Hear how Alexa will speak a response entered in plain text or SSML. [Learn more about supported SSML tags](#).

For example: Here is a word spelled out: <say-as interpret-as="spell-out">hello</say-as>.

hello world

4. Test it. All “Starter Code” and “Part 1” local unit tests should pass. Try the skill out by uploading your changes to the Interactive model in the Developer Portal and AWS Lambda.

Part 2: Add a feature

In addition to the **GetNewFactIntent** intent already included, add an **intent** including a built-in **slot** named **FACT_YEAR** that will provide the user with a fact matching the year requested. Name this intent **GetNewYearFactIntent**. Built-in Amazon slots can be used for the year. Consider using **AMAZON.FOUR_DIGIT_NUMBER** for this purpose. This is not required, however, if you prefer to try a different slot definition.

1. Provide at least 15 utterances for the new intent.

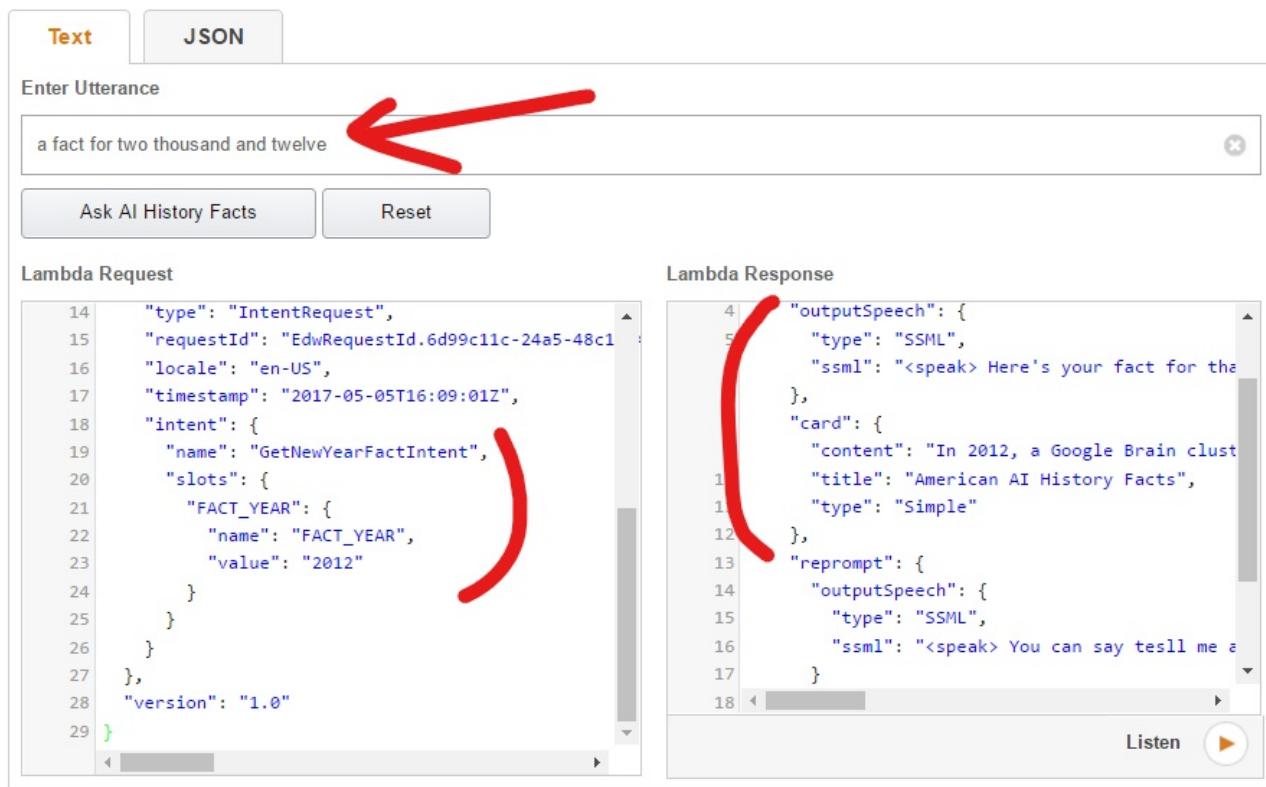
[Preview] Project: Alexa History Skill

3. Test it. All “Starter Code”, “Part 1”, and “Part 2” local unit tests should pass. Try the skill out by uploading your changes to the Interactive model in the Developer Portal and AWS Lambda.

Part 3: Add conversational elements

As discussed in the videos, conversations with a VUI will seem more natural if the session window can be left open to continue request/response interactions. Additionally, adding randomization to the Alexa response text provides a more natural feel to the conversation.

1. Change your “tell” statements to “ask” statements as directed by the TODO’s in **index.js** and include reprompt messages as necessary.
2. Change the **GET_FACT_MESSAGE** snippet to an array of at least 5 similar phrases. Randomize this portion of the Alexa response.
3. Test it. All local tests should now pass. Deploy your changes to the Interactive model in the Developer Portal and AWS Lambda.
4. Provide a screen capture (PNG) from the Service Simulator in the Developer portal of your skill working. The screen capture should include the part of the Lambda Request that shows the **GetNewYearFactIntent** and **slot** with the year requested. The Lambda Response side only needs to show that a fact was provided. Note that in order to request a slot with the Simulator, you will need to phonetically request a year. For example, if the year is 2012, the input will need to be “two thousand and twelve” rather than “2012” in the simulator. Save the screen capture for submission with the name **skill_simulator.png**



The screenshot shows the Alexa Service Simulator interface. At the top, there are 'Text' and 'JSON' tabs, with 'Text' selected. Below is an 'Enter Utterance' field containing the text 'a fact for two thousand and twelve'. A red arrow points from this field to the 'Lambda Response' section. The 'Lambda Request' section shows the following JSON:

```

14  "type": "IntentRequest",
15  "requestId": "EdwRequestId.6d99c11c-24a5-48c1
16  "locale": "en-US",
17  "timestamp": "2017-05-05T16:09:01Z",
18  "intent": {
19    "name": "GetNewYearFactIntent",
20    "slots": {
21      "FACT_YEAR": {
22        "name": "FACT_YEAR",
23        "value": "2012"
24      }
25    }
26  },
27  "version": "1.0"
28
29

```

The 'Lambda Response' section shows the following JSON:

```

4  "outputSpeech": {
5    "type": "SSML",
6    "ssml": "<speak> Here's your fact for the
7    },
8    "card": {
9      "content": "In 2012, a Google Brain clust
10      "title": "American AI History Facts",
11      "type": "Simple"
12    },
13    "reprompt": {
14      "outputSpeech": {
15        "type": "SSML",
16        "ssml": "<speak> You can say tell me a
17      }
18

```

At the bottom right of the response section is a 'Listen' button with a play icon.

Optional Additional Testing

In addition to testing with unit tests and the Service Simulator, you may find it useful to try your skill with one or more of the following:

[Preview] Project: Alexa History Skill

- ~~Amazon Echo, Echo Dot, Echo Tap devices. If the device is on the same account as the development code, you can "open" the skill there.~~

NEXT