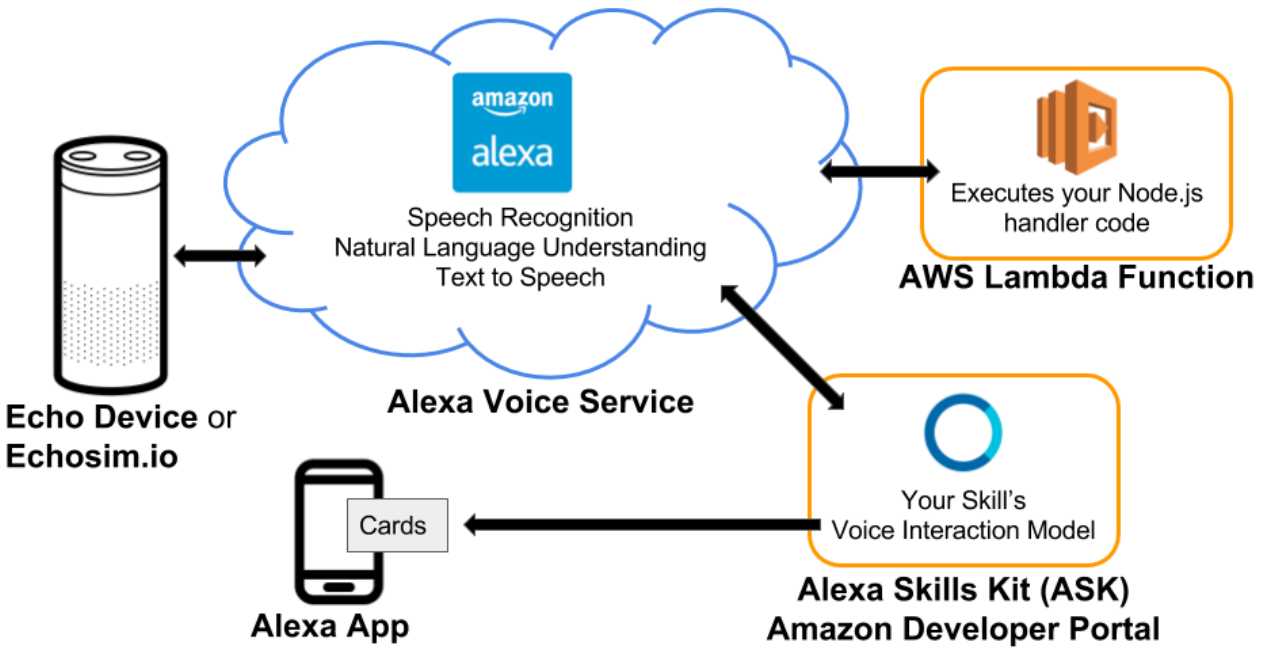
[Preview] Project: Alexa History Skill

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.

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 Lamda
  - **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

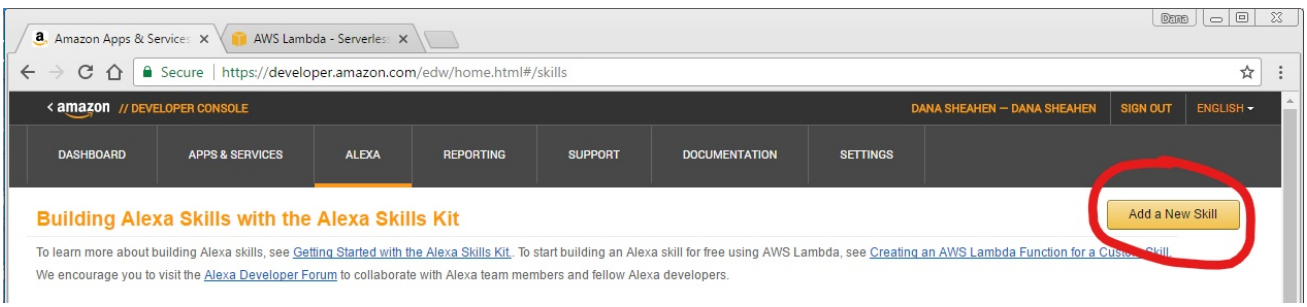
- Deploy the starter code to verify that it works with your accounts in its simple form. This is the same process you went through with the Space Geek Lab. If you need a refresher, step-by-step instructions are provided here.

Deploying your Skill

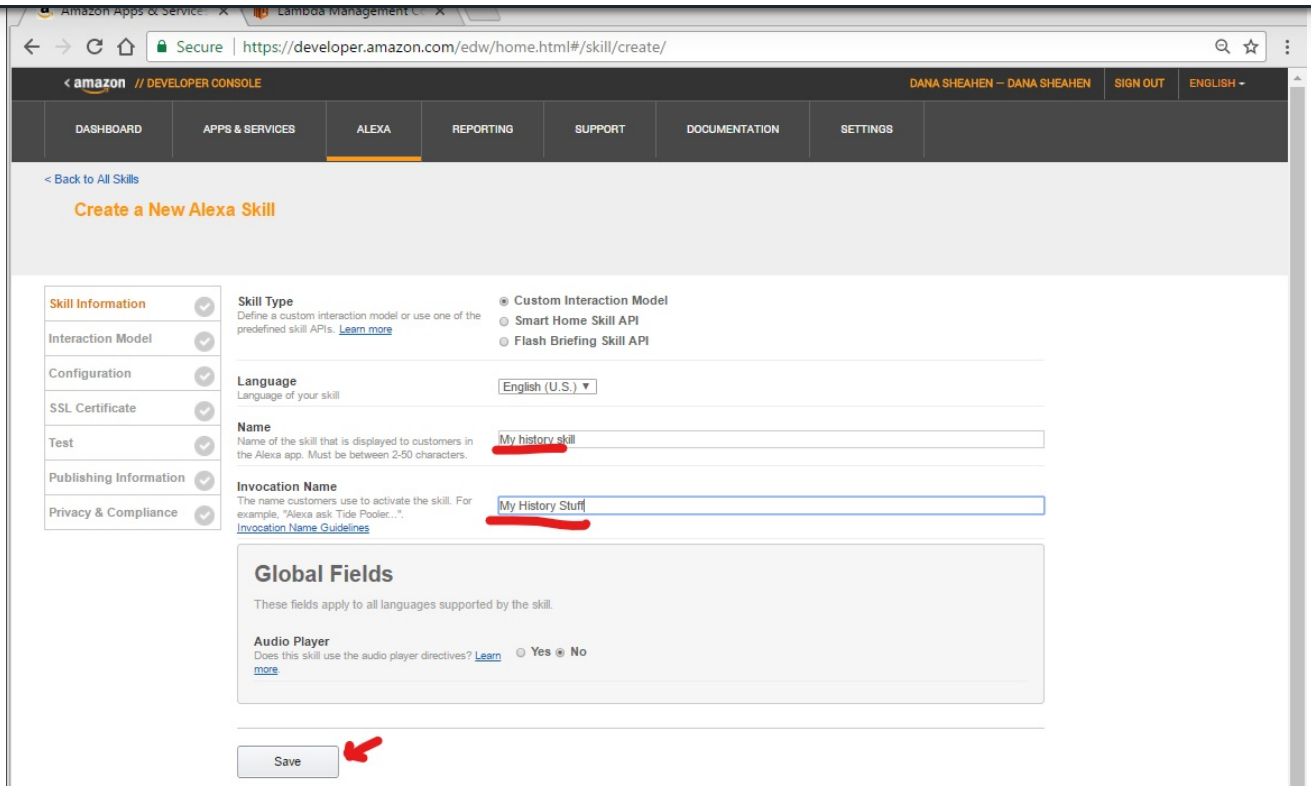
The Space Geek Lab steps you through setting up an Alexa Skill. The starter code can be deployed in the same way. The following step-by-step instructions assume you already have completed the steps in Getting Started, and are ready to deploy your starter code.

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":

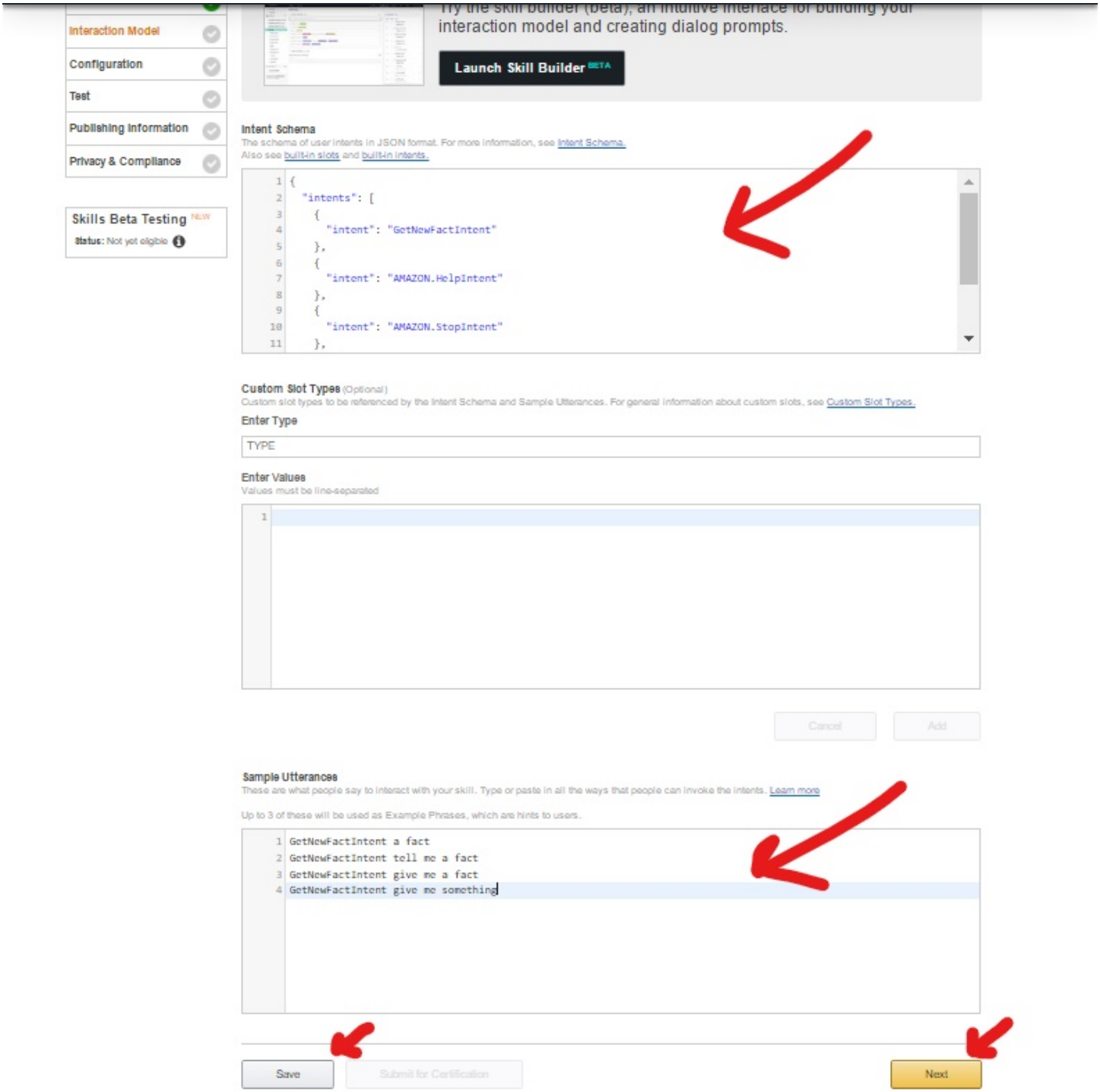


[Preview] Project: Alexa History Skill



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".

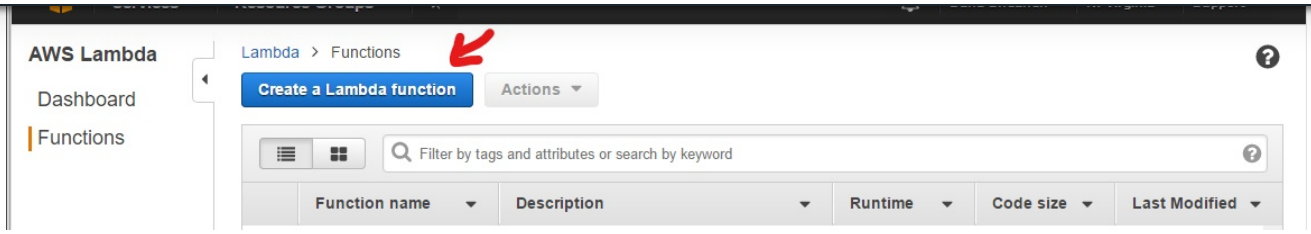
[Preview] Project: Alexa History Skill



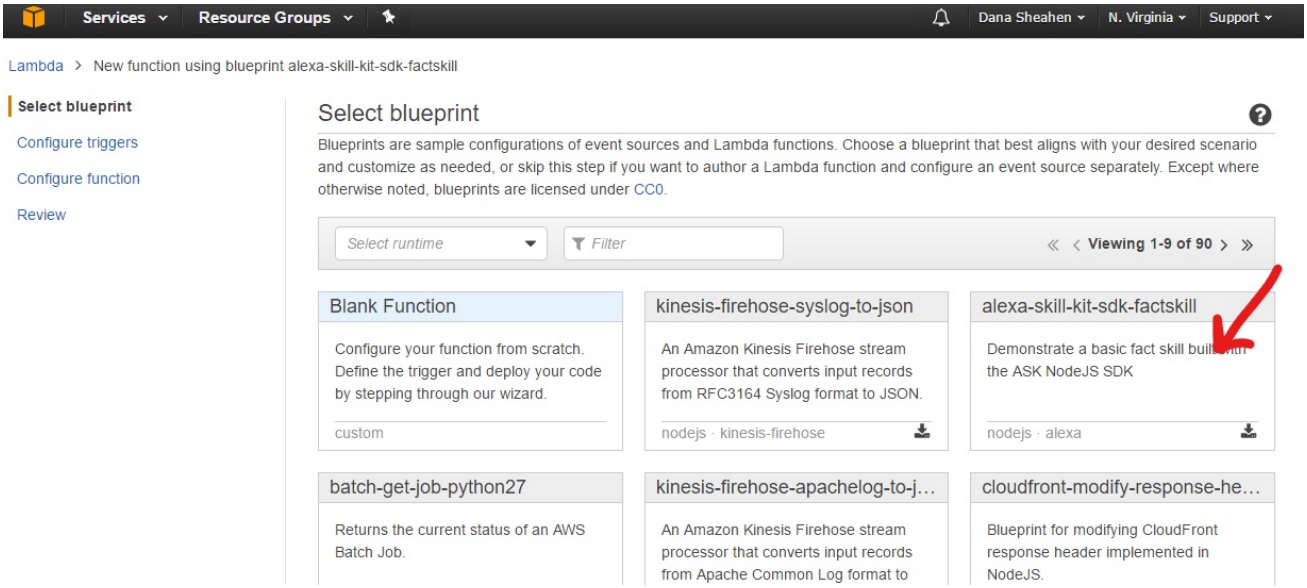
5. From the src directory of your starter code, zip index.js, facts.js, and the node_modules directory together. This zip file will be uploaded next to AWS Lambda.

6. Navigate to AWS console and choose the AWS Lambda Service. Click “Create a Lambda Function”.

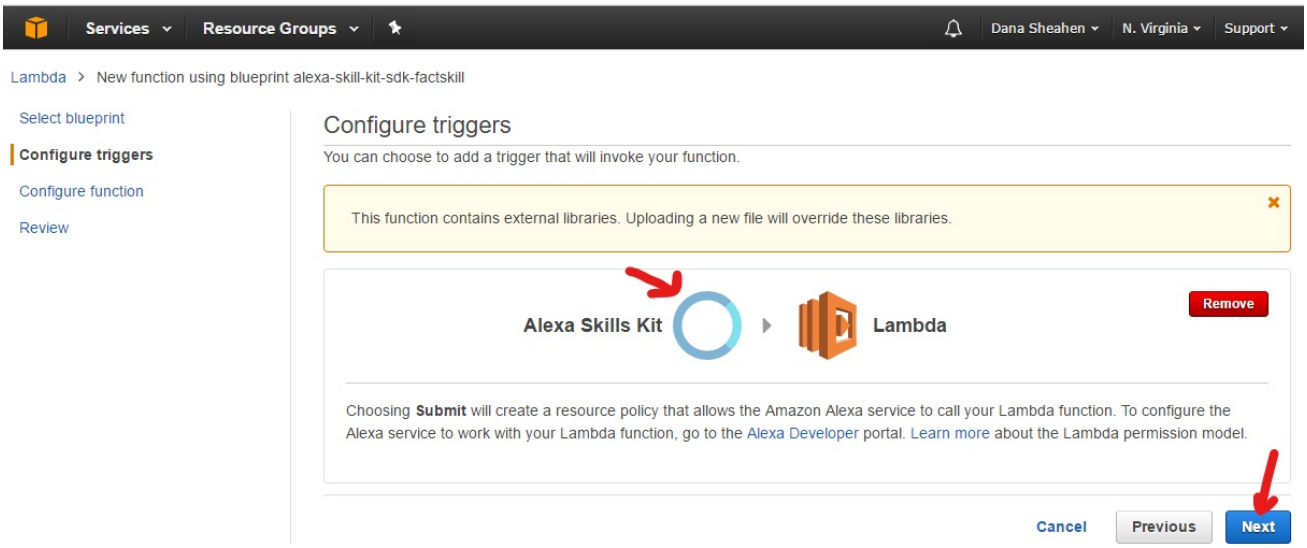
[Preview] Project: Alexa History Skill



7. Select the “alexa-skill-kit-sdk-factskill” blueprint.



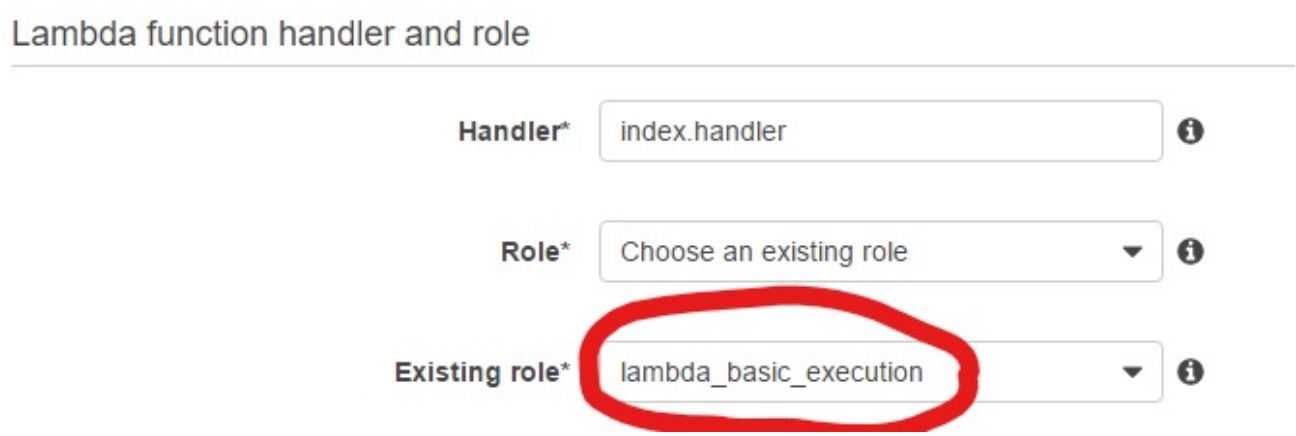
8. Click the gray box to configure the trigger (the skill we defined in the Developer portal) and choose “Alexa Skills Kit” from the drop down menu. Click “Next”.



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

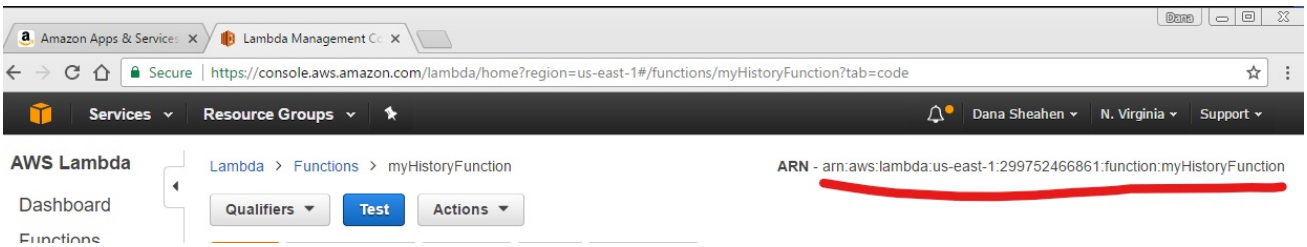
[Preview] Project: Alexa History Skill

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.

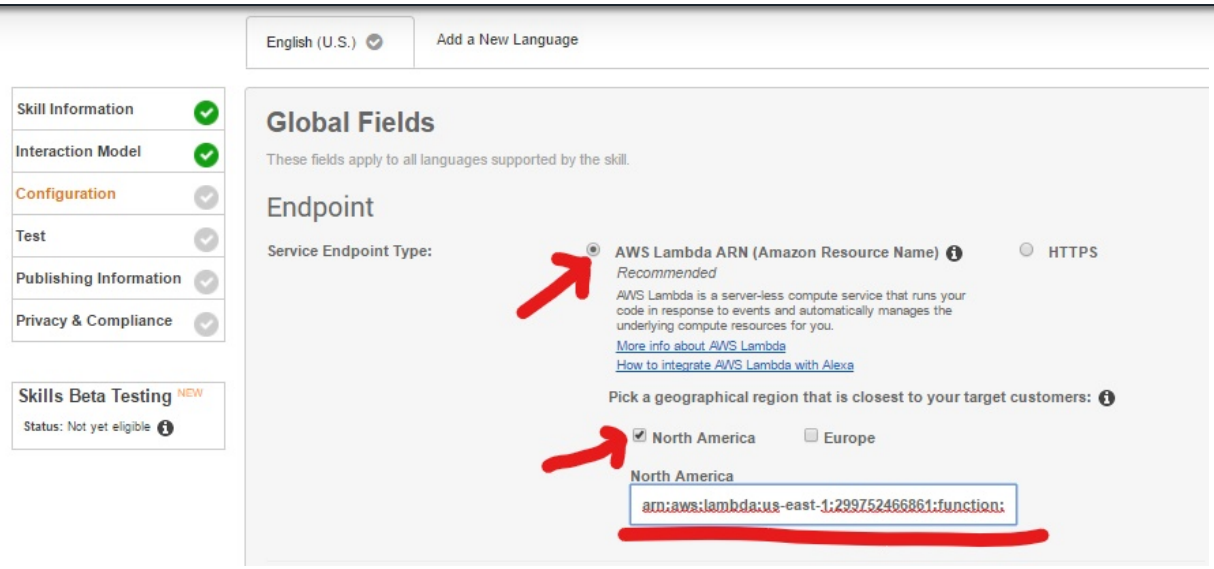


11. On the Review page, click "Create function". You should get a new page that includes a "Congratulations!".

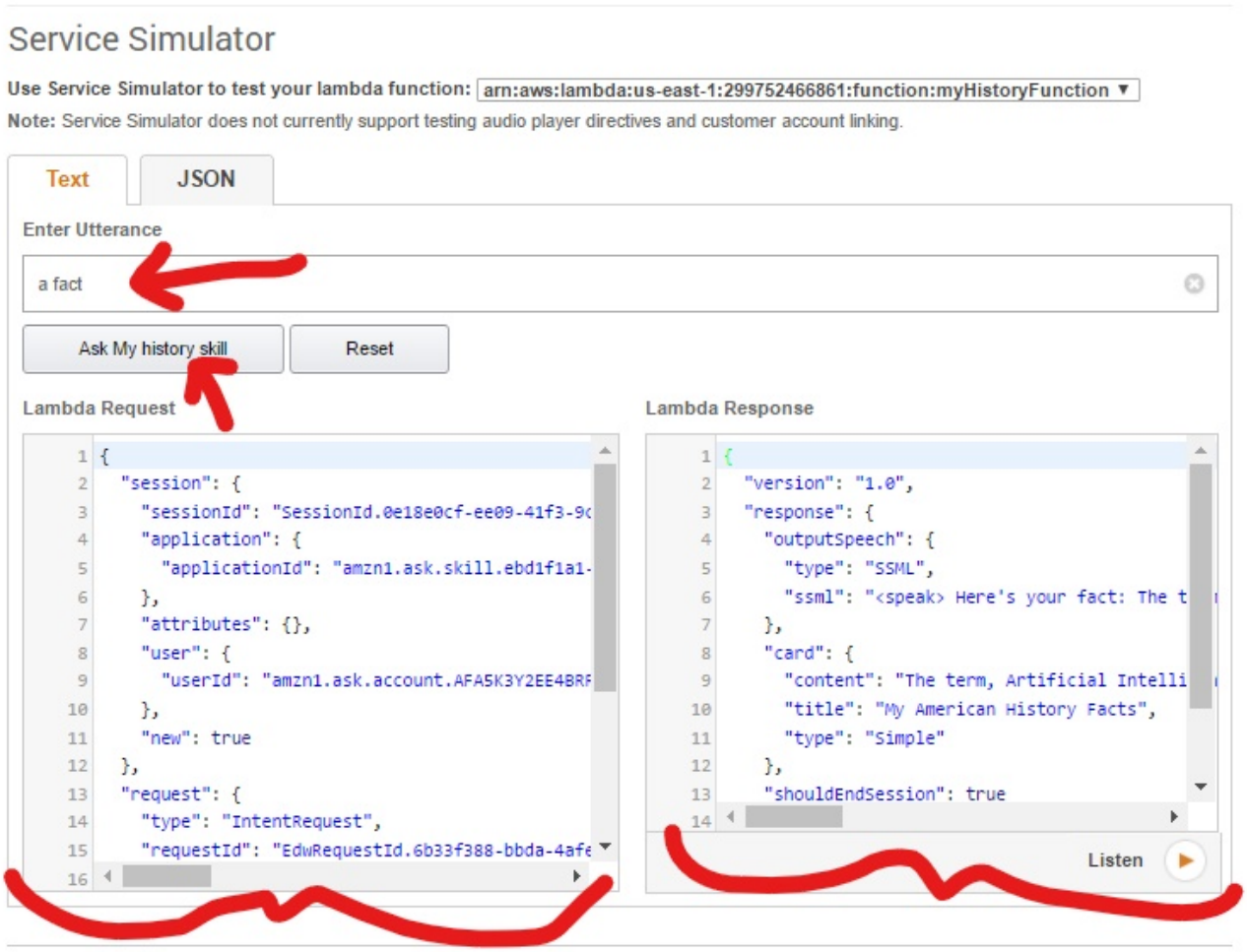
12. Upload the zip file you created earlier with the starter code files. Do this by selecting the "Code entry type" drop down menu as "Upload a ZIP file". Click the "Upload" button to find your file and then click "Save".



[Preview] Project: Alexa History Skill



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.



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.

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:

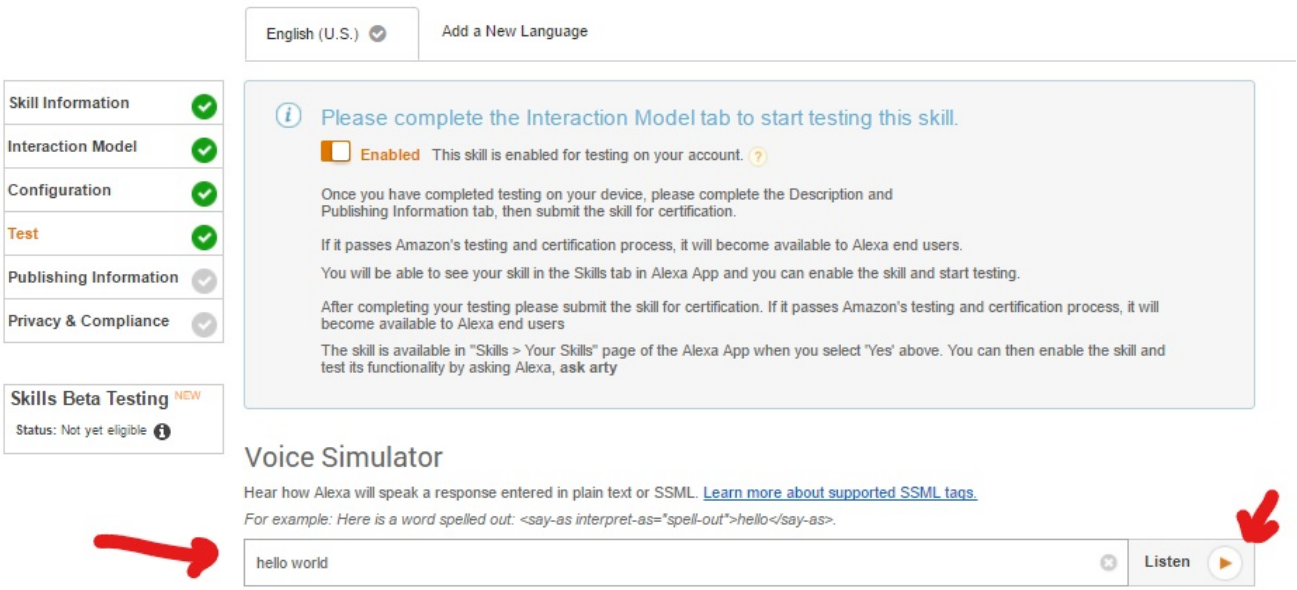
1. Customize the fact skill
2. Add a feature using an additional intent and a slot
3. Add conversational elements

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:



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

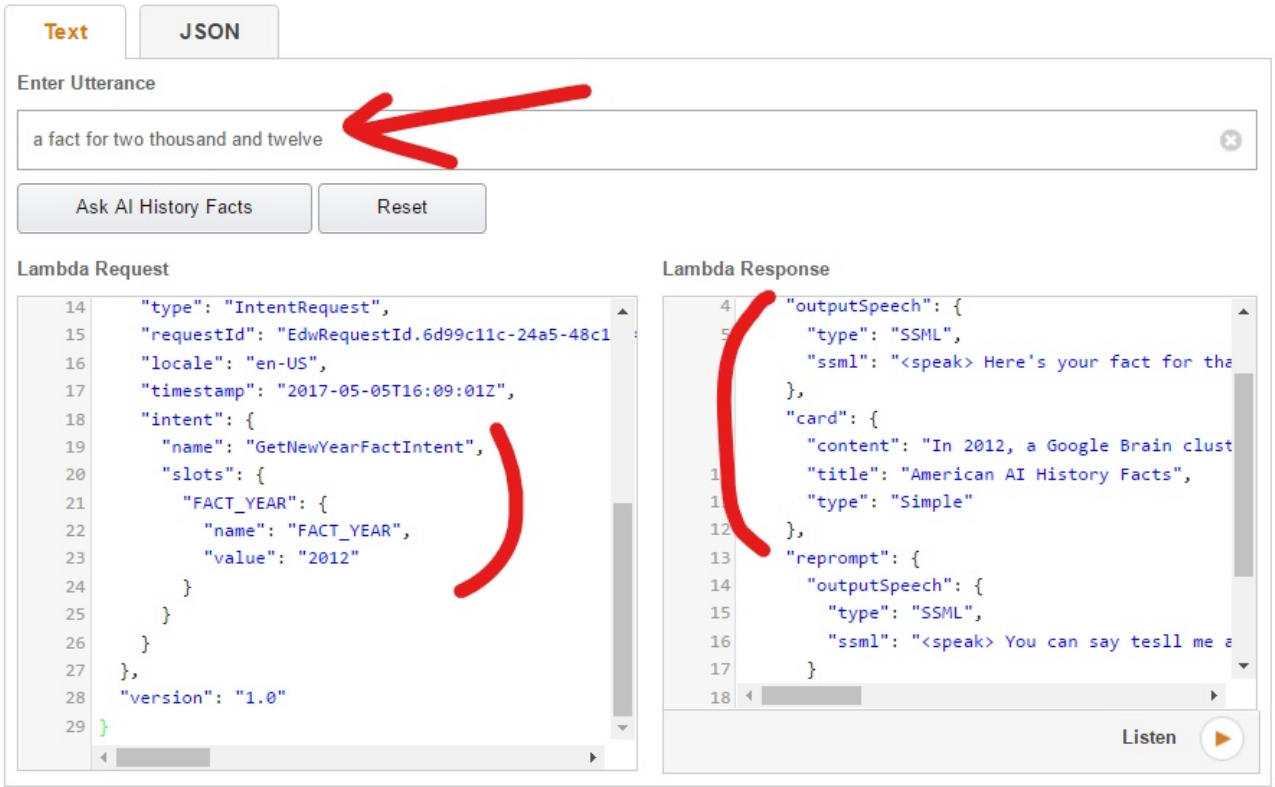
As discussed in the videos, conversations with a VUI will seem more natural if the session window can be left open to continue request/response interactions. Additionally, adding randomization to the Alexa response text provides a more natural feel to the conversation.

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`



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