PRE-HACKATHON ALEXA CHATBOT

This document guides in building the custom Amazon ALEXA for the intent "Zodiac Sign." with Two different approaches.

Index

1st Approach :

Note:

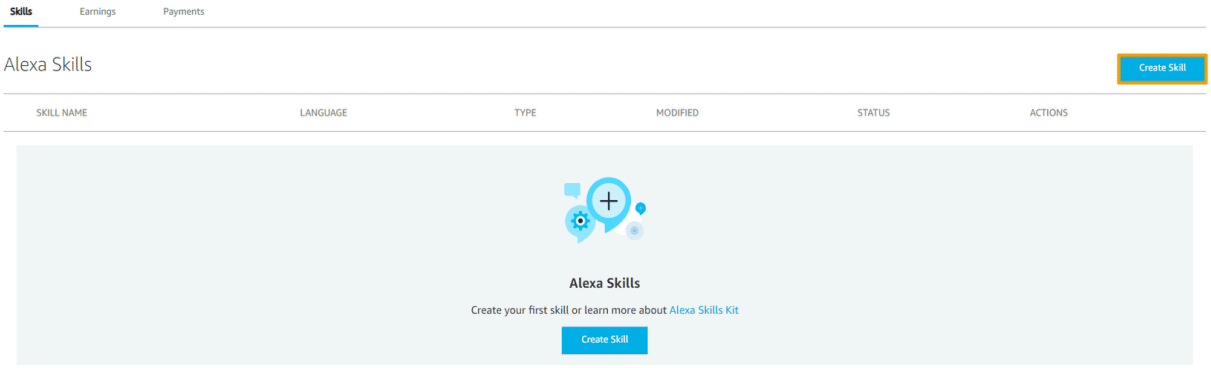
- The steps outlined above will guide you through the process of creating a zodiac sign intent in Alexa console.
- Repeat steps 3-4-5 to build another intent in the same skill.

- Alexa handles switching between the multiple intents.
- For each intent, a minimum of 50 utterances shall be provided with 3 slots and the respective speech prompts (slot dialogs).

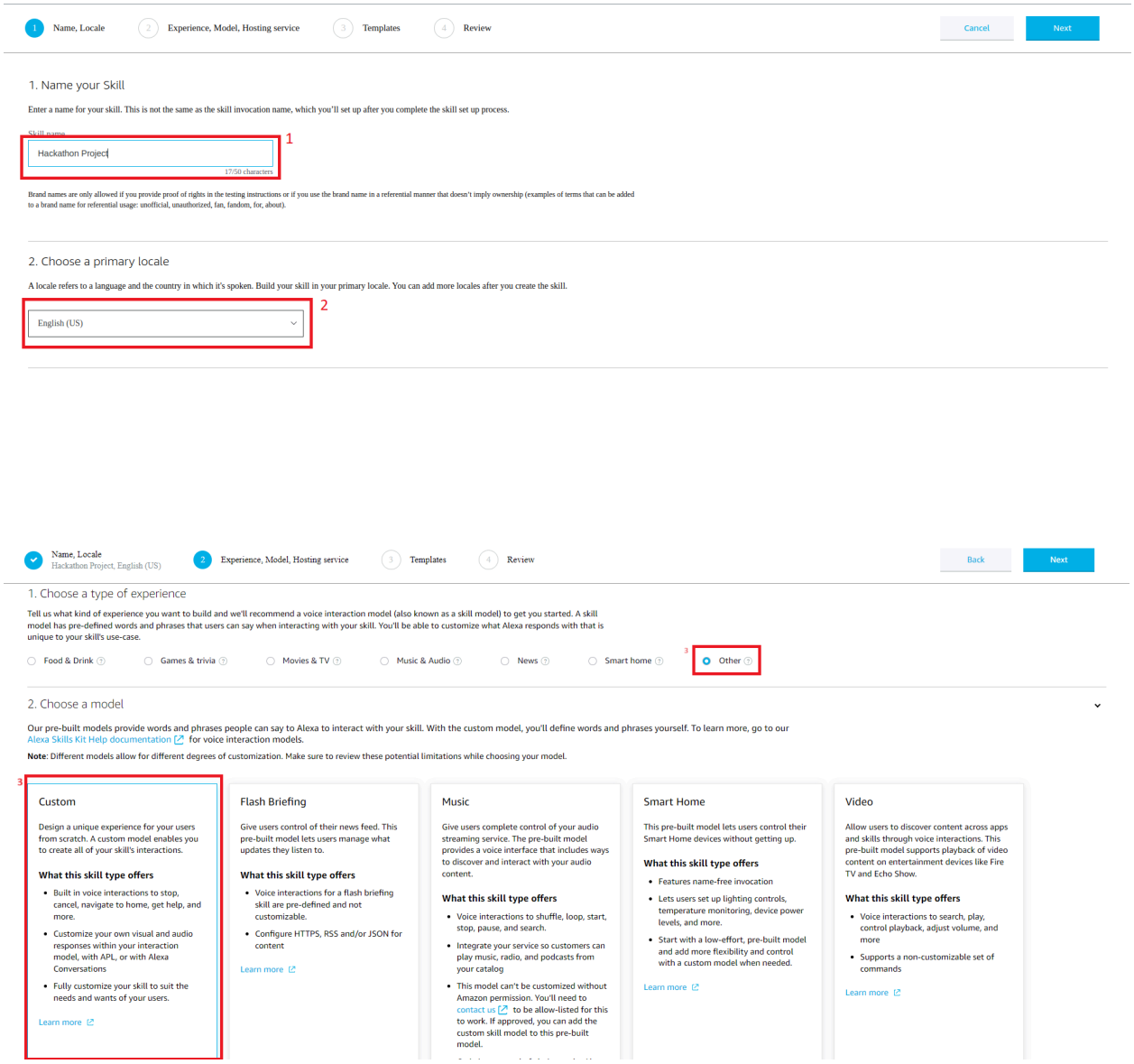
Step1: Login

To get started, log into the Alexa developer console with your Amazon Developer account. If you do not have an account, click here to create one and then login to the developer console.

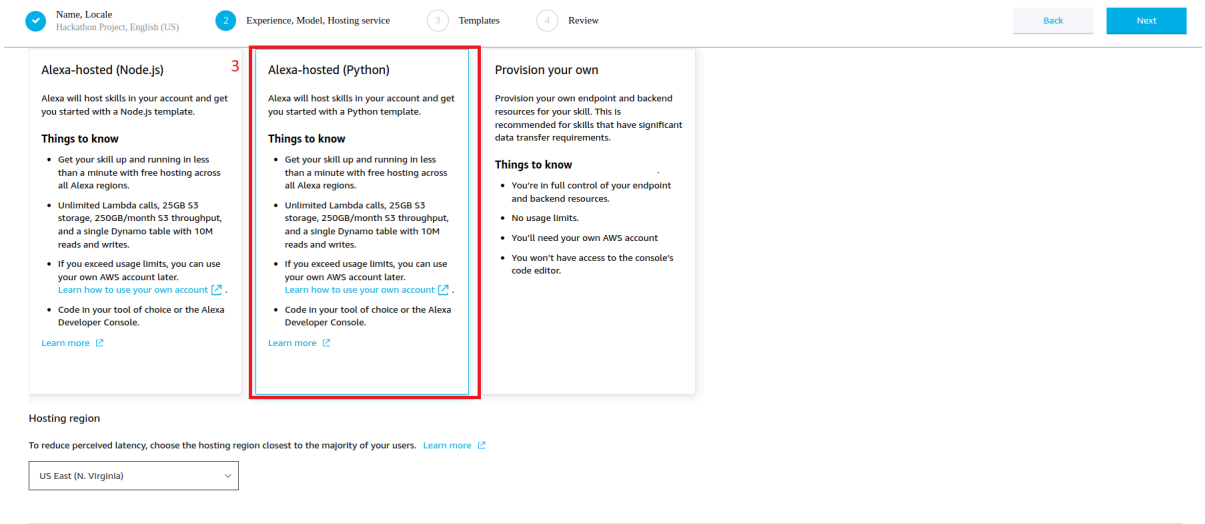
Step2: Create your skill



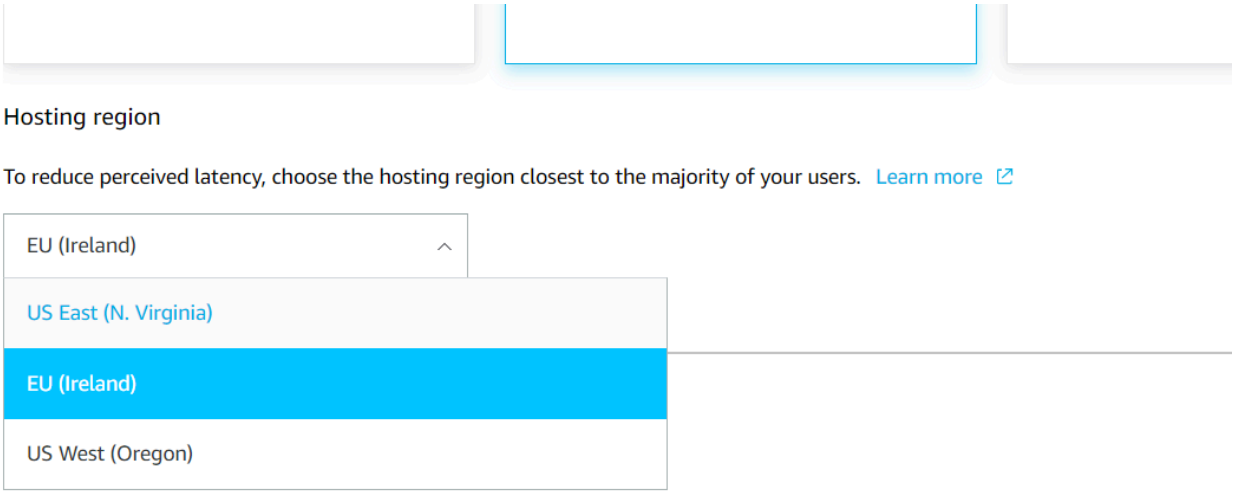
- Click Create Skill on the right-hand side of the console. A new page displays.
- Enter your preferred name of skill (e.g. Hackathon Project) In the **Skill name** field and leave the **Default language** set to **English (US)**.
- You are building a custom skill. **Choose a model to add** to your skill, select **Custom** and **choose a method to host your skill's backend resources**, select **Alexa-Hosted (Python). Refer below image.**



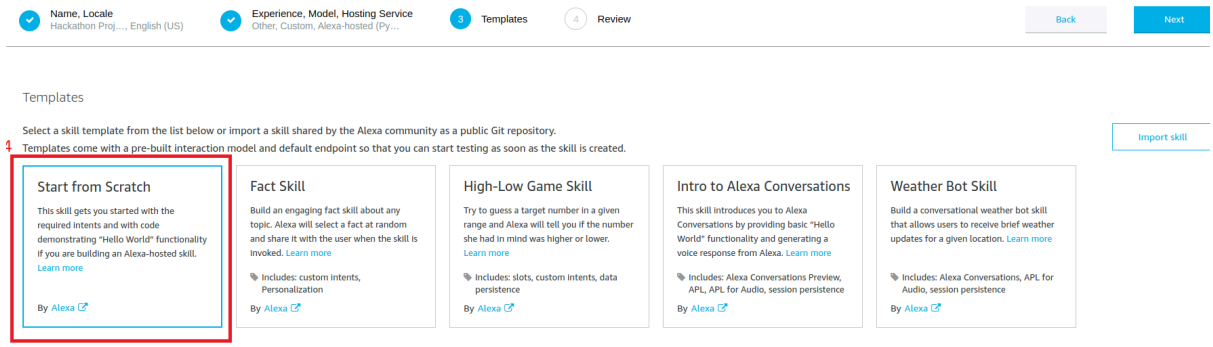
- Choose a type of experience as other and choose a model as custom since we are building our custom model and scroll down.



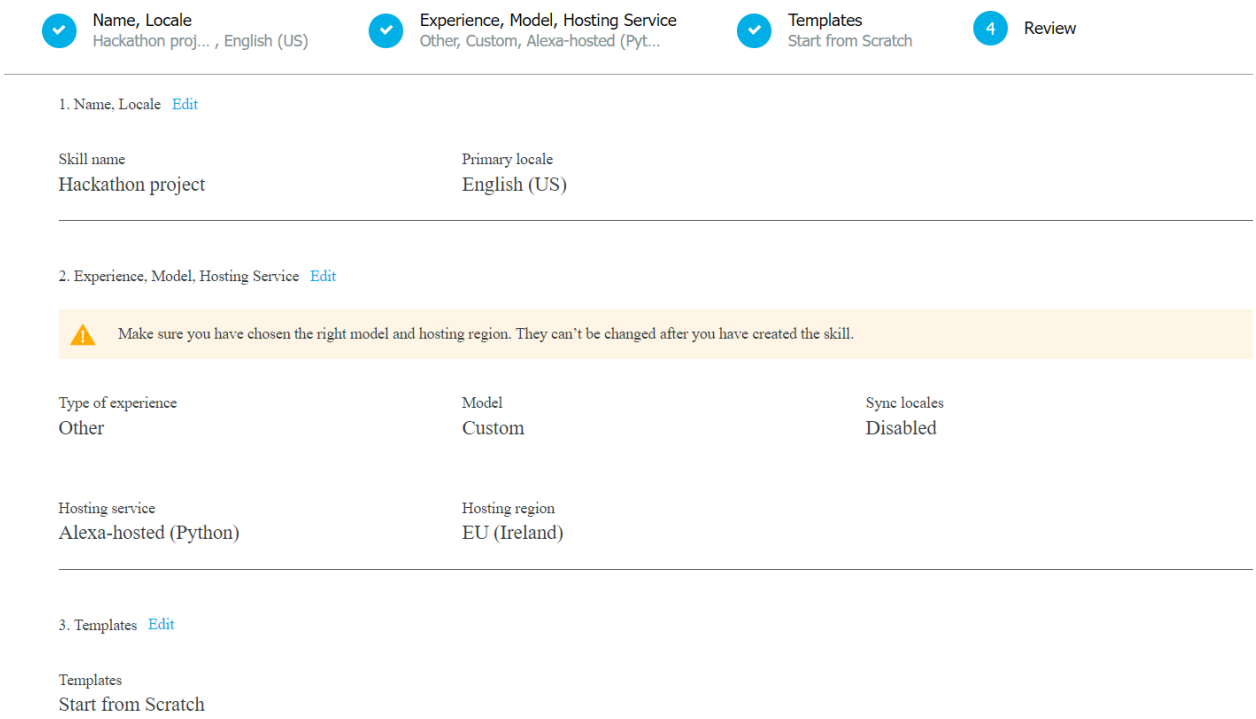
- Scroll down and select Alexa-hosted (python) and hosting region EU (ireland) as below.



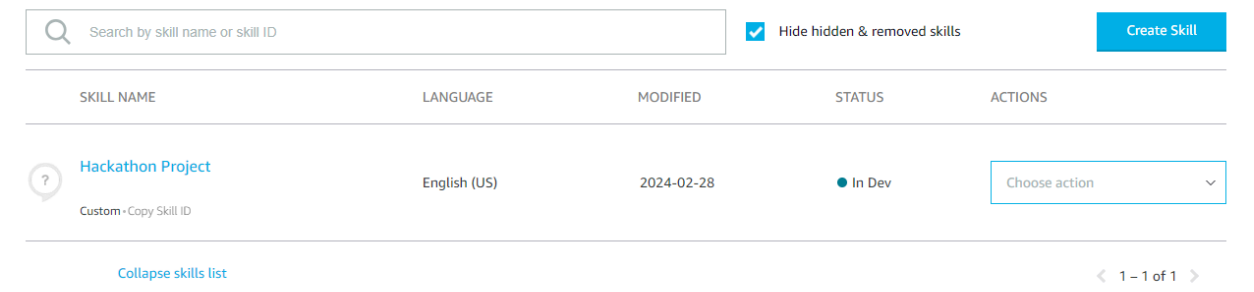
- Choose a template to select a Hello World Skill From the given Templates and at the top of the page, Click Choose as per the below image.



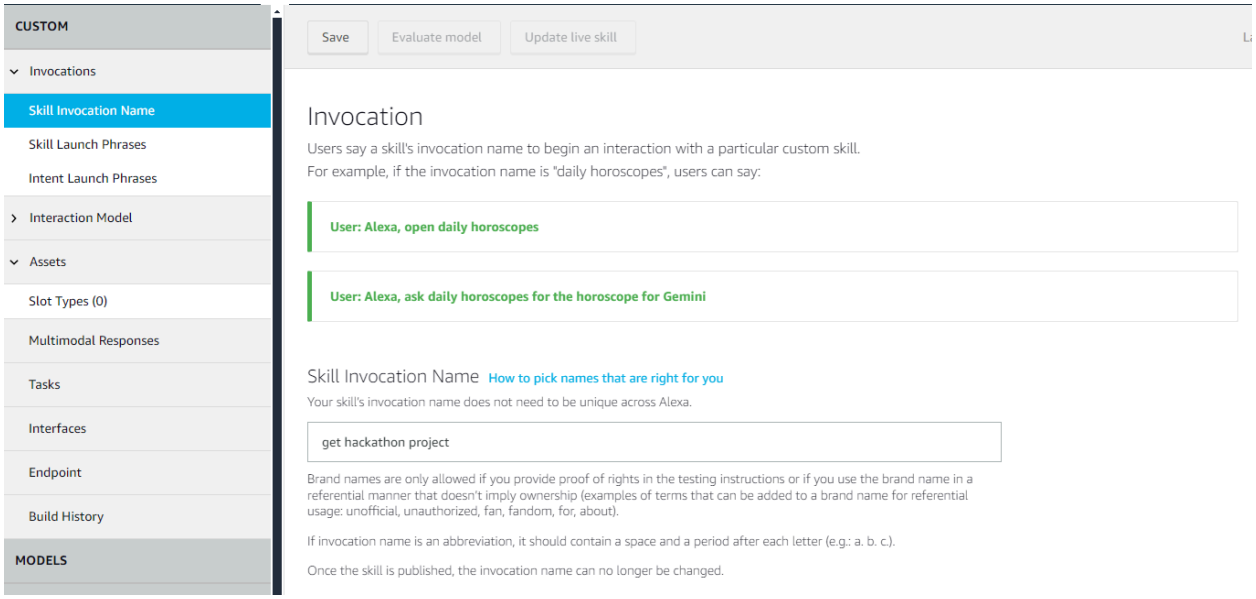
- Click on next you will be able to see the below details and verify.



- Click on Create skill and It takes a few moments for AWS to provision resources for your skill. When this process completes, move to the next section.
- **Note:** When you exit and return to the [Alexa developer console], find your skill on the Your Skills tab. In the Alexa Skills list. Click on choose action and select Edit to continue working on your skill as per the below image.



- Click on the Build tab and change the invocation name as per your skill as shown in the below image. As this invocation name will be used in testing.
  Note: Invocation is the act of beginning an interaction with a custom skill, for example the phrase: "Alexa, get hackathon project" tells Alexa to use the hackathon project skill. The keyword "get hackathon project" is known as invocation name *(you can give any name for the skill or invocation)*.



Step 3: Create intent and slots to capture information

In this section, you will make the skill more useful by having it ask the user for their date of birth. So that when the user responds, the skill will understand and repeat the user's zodiac sign back to them.

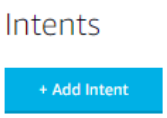
To do this, you will need to use **utterances, intents, and slots**. You will also learn how to use **dialog management** to have your skill automatically ask follow-up questions to collect the

required information. **For example**, if the user says, "**I was born July 12th**," dialog management will automatically ask the user **what year they were born**.

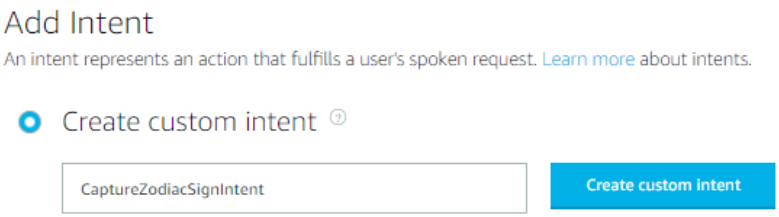
At the end of this module, your **Zodiac Sign** intent will be able to ask the user a question and listen for the answer further to respond to the user.

Follow the Below Steps to Build your Model:

- Click on Intents tab under the Interaction Model portion and then click on + Add Intent button as shown below. The Add Intent window opens



- Choose **"Create custom intent"** and enter the Intent name for your skill (Eg. **CaptureZodiacSignIntent**), furthermore click Create custom intent. The intent will be created.

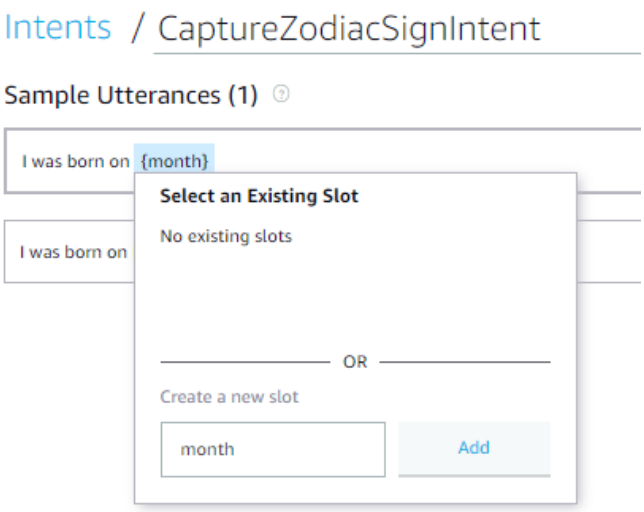


- **Note:**
  - An **intent** represents an action that fulfills a user's spoken request. When you create a new custom intent, you provide a name and a list of utterances that users would say to invoke this intent
  - The **sample utterances** are set of likely spoken phrases mapped to the intents. An utterance is what invokes the intent. In response to the birthday question, a user might say - "I was born on November seventh, nineteen eighty-three." You will add this utterance to the CaptureZodiacSignIntent by typing it in exactly the way the user is expected to say it.
  - **Slots** are a very powerful accessory for building a custom Alexa skill. For example, the statement "I was born in {month}" means that the user can provide any month to our skill from the slot month.

- In the Sample Utterances field, type the following, and then press ENTER or click the + icon: **I was born on November seventh nineteen eighty three** Notice that the text does not include **punctuation.**

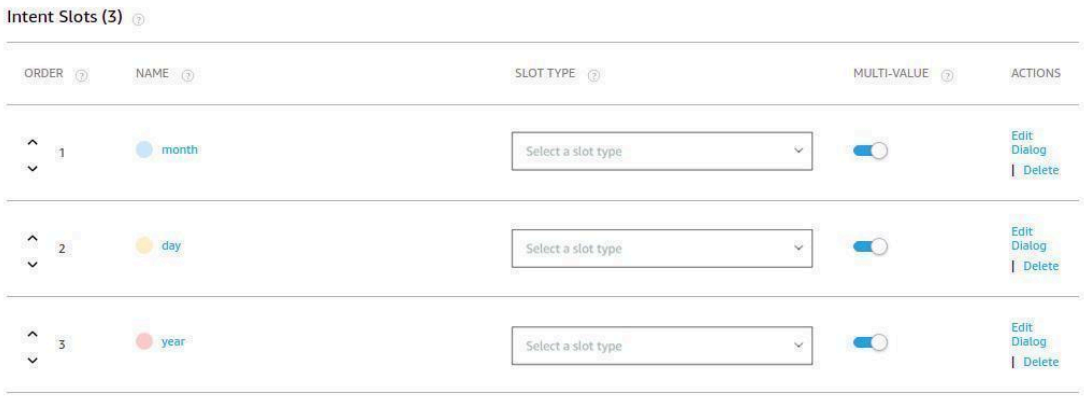
- From this utterance, there are three key pieces of information to collect: **month, day, and year**. These are called **slots.** You need to let Alexa know which words are **slots and what kind of slots they are.**

- Start with the **month slot**. In the **utterance**, you will replace the word representing the **month (November)** with the word month in **curly braces ({ })**. This creates a slot called **month**. The utterance will then look like this: **I was born on {month} seventh nineteen eighty three.** Follow the below steps to create a **slot**.
    - Select the word in the sample utterance where the slot should go and type the name of the slot in curly braces (**for example, {month}**).

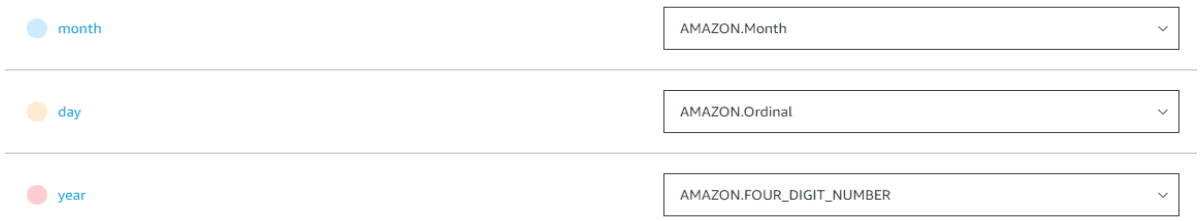


    - Repeat this process for the other variable pieces of information (day and year). Your utterance should now look like this: I was born on {month} {day} {year}
    - Like, In the Sample **Utterances** field, type **{month} {day} {year}**, and then press **ENTER** or click the + icon.

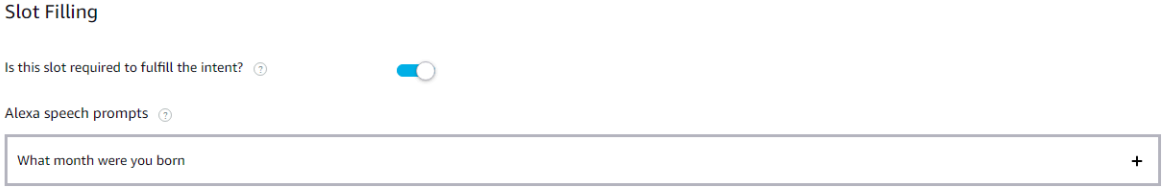
- Scroll down the page to **Intent Slots**. This area displays the three slots you have created for this skill.



- Slots are assigned from the Slot Type drop-down menu to the right of each slot.
- There are two types of slot types: **custom and built-in**. Wherever possible, use built-in slots. Alexa manages the definitions of built-in slots. These slots begin with AMAZON followed by what they define (for example, AMAZON.Month).
- Here you can find the built-in slot types a available in alexa List of [Amazon Slots]
- If an applicable built-in slot does not exist, **create a custom slot** in the "*Slot Types (0)"* section and add all the values it represents.
- To the right of the month slot, select AMAZON.Month from the Slot Type drop-down menu. For the day as select AMAZON.Ordinal and year slots, as AMAZON.FOUR_DIGIT_NUMBER respectively as the slot type.

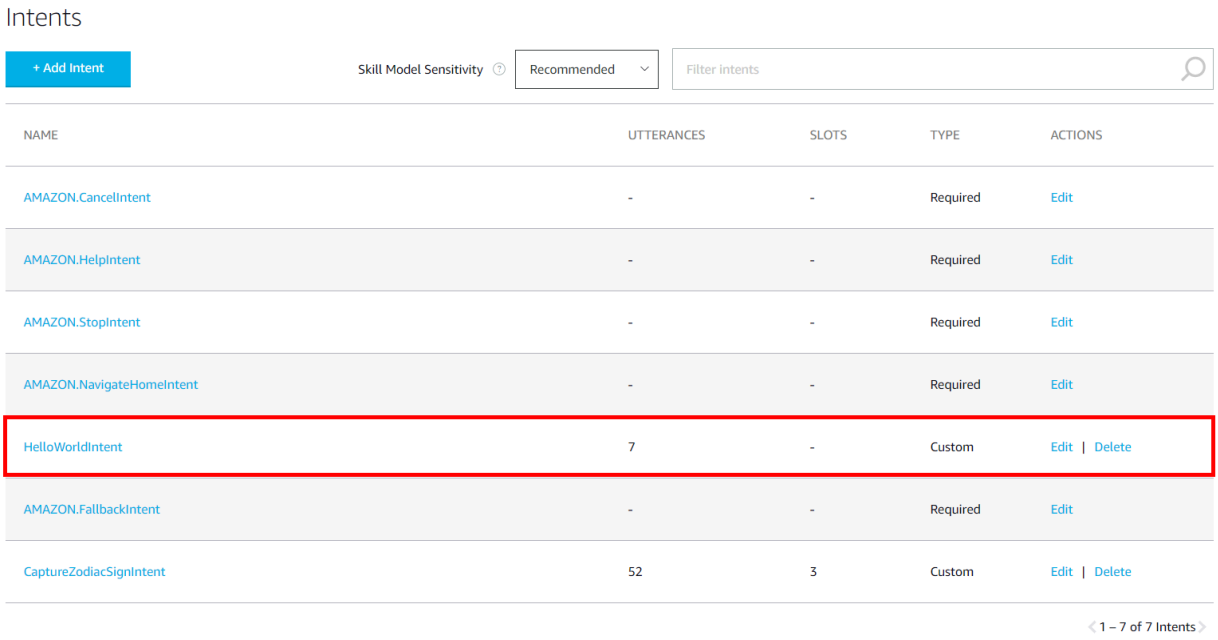


- Under ACTIONS click **Edit Dialog**, toggle to make the **slot required.**
- In the Alexa speech prompts field, type "**What month were you born"** and then press **ENTER** or click the **+ icon,** you can also provide User utterances in the next field.

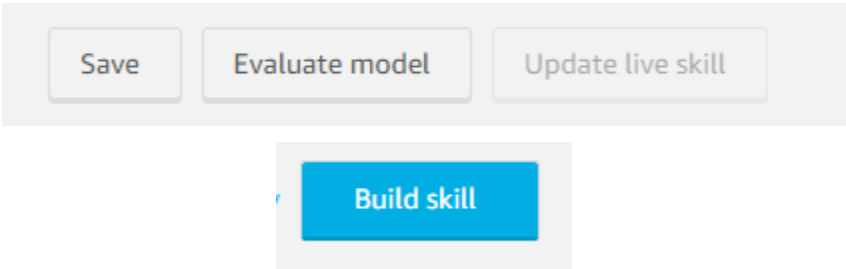


- Repeat the process for the **day** and **year slots.**

- Delete the **HelloWorldIntent** by clicking delete button to the right of it. When prompted, click **Delete Intent**.
- Be careful to delete **HelloWorldIntent** and not **CaptureZodiacSignIntent.**



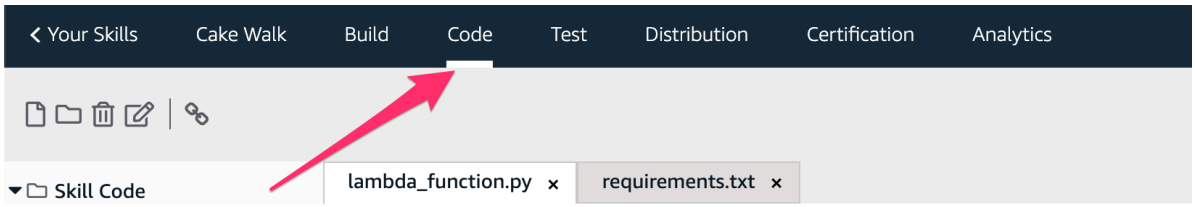
- At the top of the page, click **Save** and **Build Skill.**



- When you click **Build Skill**, your skill starts to build the training data that will help Alexa know how to map what the user says to your skill's intents. It may take a minute for the model to build.

Step 4: Let's Start Building the Model in the Backend.

In this step, you will update your backend code to respond to the user when they open the skill.



Click the **Code tab**, you can see an online code editor with some files already set up for you to get started. In particular, you'll see the following three files in the lambda sub-directory:

- **lambda_function.py:** This is the main entry point of the backend service. All the request data from the Alexa intent is received here and is supposed to be returned from this file only.
- **requirements.txt:** This file contains the list of Python packages that are being used in this project.
- **utils.py:** This file contains some utility functions required for the lambda function.

Within the **lambda_function.py**, to define how your skill responds to a request, you will define a handler for each intent. There are two pieces to a handler:

- **can_handle()** recognizes each incoming request that alexa receives.
- **handle()** returns an appropriate response.

- The **speak_output** variable contains the string of words the skill should say back to the user when they launch the skill.

What should happen when a user launches the Zodiac Sign Skill?

In this case, you want the skill to simply confirm that the user opened it by saying, **"Hello! Welcome to Zodiac Signs. When were you born?"**

- Within the **LaunchRequestHandler** object, find the **handle()** function, and the line that begins **speak_output**. Replace that line with the following:

```
speak_output = "Hello! Welcome to Hackathon Project. Would you like to now your zodiac sign?"
```

- Within the LaunchRequestHandler, you will find the .ask() function:
  - The ask() function tells the skill to wait for the user to reply, rather than simply exiting.
  - It allows you to specify a way to ask the question to the user again, if they don't respond.
  - A best practice is to make your reprompt text different from your initial speech text.
  - Perhaps for a variety of reasons the consumer did not respond. The skill will again pose the initial question but do so in a natural manner. The reprompt will provide the consumer with more information to help generate a response. Develop a new variable called reprompt text to define the reprompt text.

- Within the LaunchRequestHandler, in the handle() function, find the line that begins **speak_output** . Create a new line below it and copy and paste the following code on the new line:

```
reprompt_text = "I was born Nov. 6th, 2014. When were you born?"
```

  - Notice the reprompt gives an example of what Alexa expects the user to say by having Alexa provide her own birthday in the format she is looking for. Providing examples like this is a best practice.

- Now you want the code to pass the reprompt_text variable to the .ask() function.
- Within the LaunchRequestHandler, in the handle() function, replace .ask(speak_output) with .ask(reprompt_text)

- If you look at the code below, you will notice the HelloWorldIntentHandler. But you deleted the HelloWorldIntent, right? Not entirely. The intent is gone from the front end, but the backend handler is still there. You need a new handler, so make things easier and reuse this handler for a new one called CaptureZodiacSignIntentHandler.
- Find the line that starts the class HelloWorldIntentHandler. On that line, rename HelloWorldIntentHandler to CaptureZodiacSignIntentHandler.
- Within the CaptureZodiacSignIntentHandler, on the line that begins return ask_utils.is_intent_name("HelloWorldIntent"), change 'HelloWorldIntent' to 'CaptureZodiacSignIntent'
- This change ensures that the can_handle() function will be invoked when a 'CaptureZodiacSignIntent' request comes through.
- Now you need to update the logic within the handler, Start by creating three variables in the handler to save the slots the skill is collecting.
- Within the CaptureZodiacSignIntentHandler, find the line that begins def handle(self, handler_input): comment the *speak_output = "Hello World!"* line and create a new line below it.
- Copy and paste the following code on the new line:

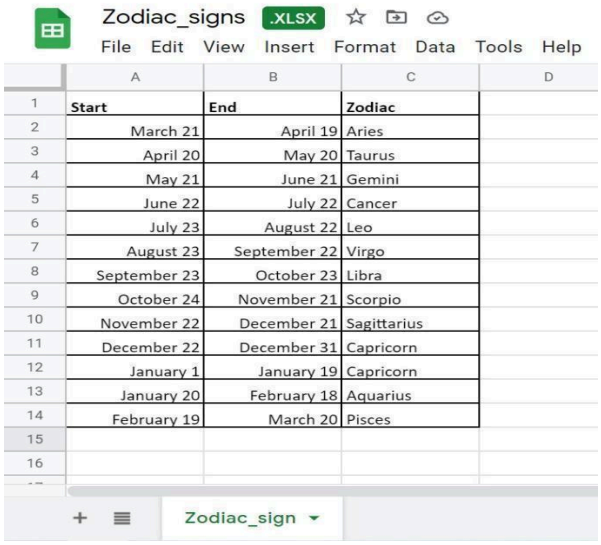
```
slots = handler_input.request_envelope.request.intent.slots
year = slots["year"].value
month = slots["month"].value
day = slots["day"].value
```

Step 5: Steps to Create Your CSV File in Google Sheets to Build an Alexa Skill with Python and Aws Lambda.

Create Your Google Sheet with Data

Let's imagine we want to create an Alexa skill that asks for my birthday and replies to me with my zodiac sign. Hence, we shall update the Google sheet with the features and labels.

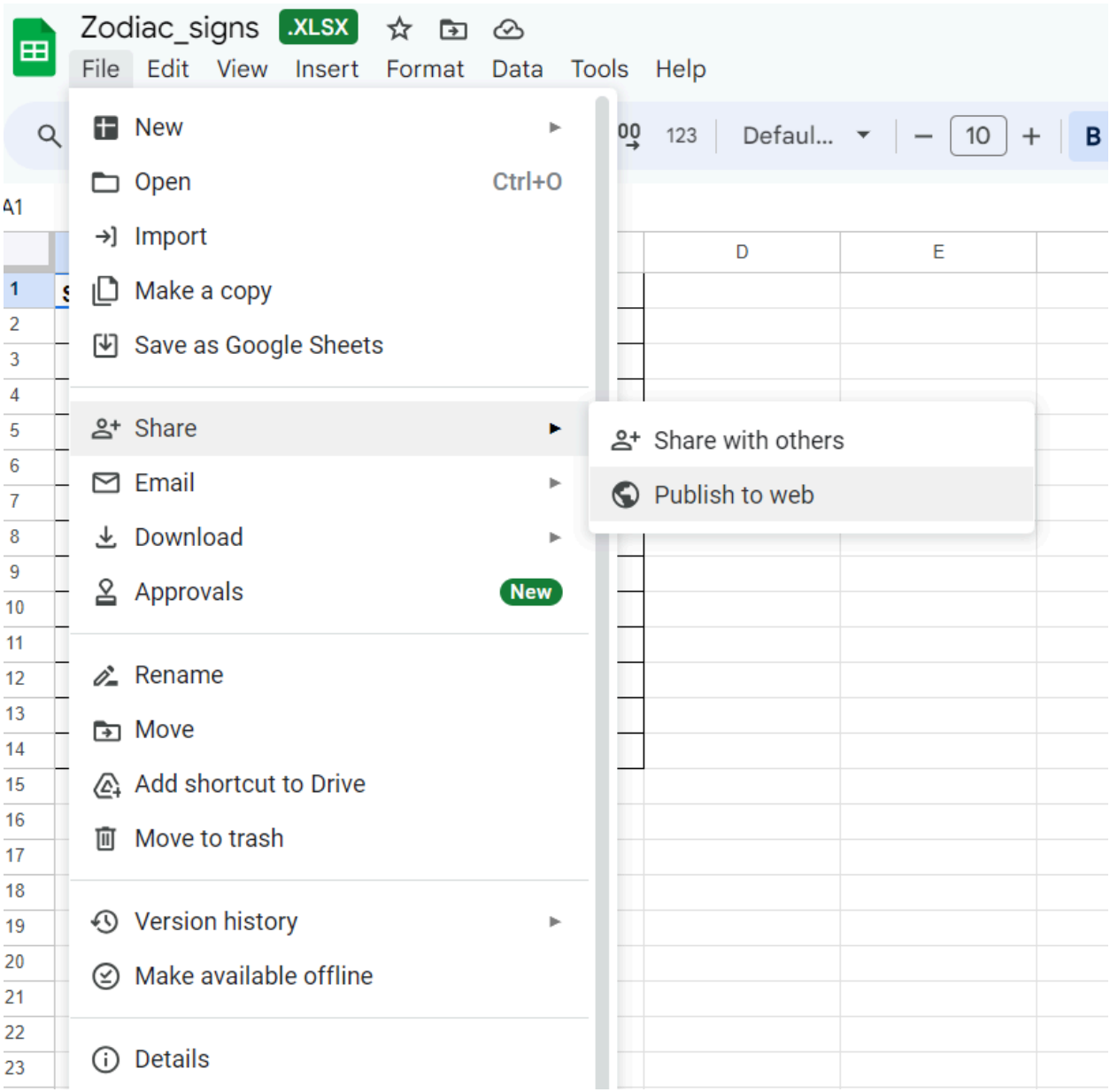
Click Here to See the Provided Zodiac Signs Csv.



	A	B	C	D
1	Start	End	Zodiac	
2	March 21	April 19	Aries	
3	April 20	May 20	Taurus	
4	May 21	June 21	Gemini	
5	June 22	July 22	Cancer	
6	July 23	August 22	Leo	
7	August 23	September 22	Virgo	
8	September 23	October 23	Libra	
9	October 24	November 21	Scorpio	
10	November 22	December 21	Sagittarius	
11	December 22	December 31	Capricorn	
12	January 1	January 19	Capricorn	
13	January 20	February 18	Aquarius	
14	February 19	March 20	Pisces	
15				
16				

Convert Your Spreadsheet into a CSV

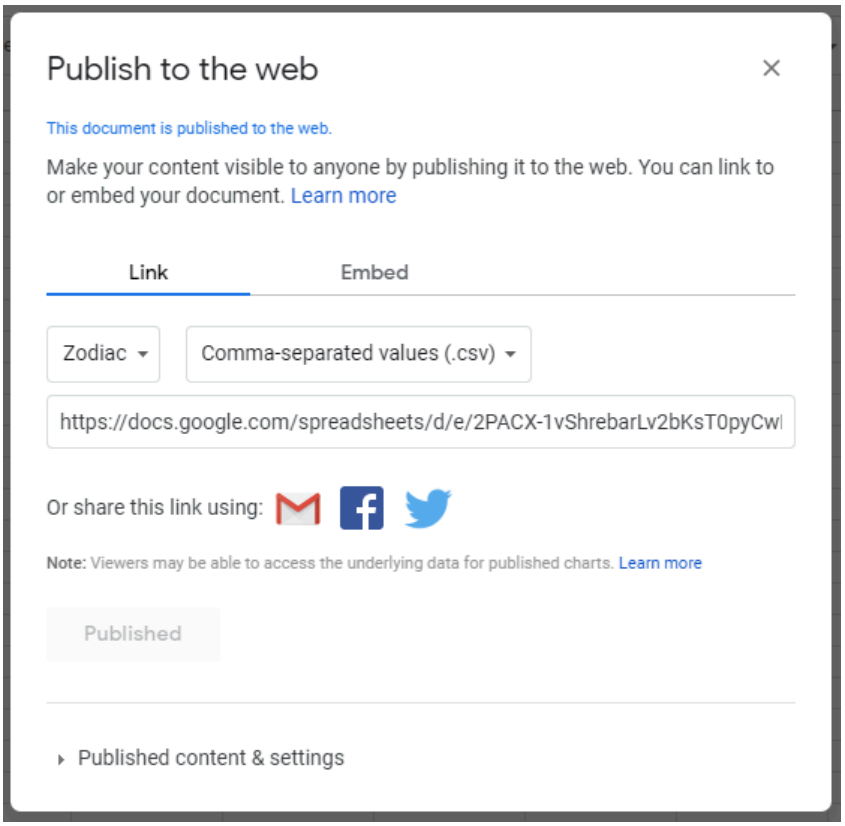
First, you need to publish your spreadsheet to the web, using File → Share → Publish To Web in your spreadsheet.



A pop up appears on clicking **"Publish To Web"**, Follow the below steps to get the link.

- From **Entire Document** dropdown → Choose a particular sheet name you are working on.
- From **Web page** dropdown → Pick **comma-separated values (CSV)**

After which your screen looks like this



Make sure you copy that link from the textbox which will be used to extract data in the process of developing your skill.

Start Editing the Alexa skill:

- Under Code Tab, Double-click the **requirements.txt** file in the pane on the left. The file opens in the editor. Create a new line and add 'pandas' at the end of the file.

```
boto3==1.9.216
ask-sdk-core==1.11.0
```

- Open **lambda_function.py** code is available in colab link . The new dependency allows you to use the CSV' to read data from the given URL. Now, you need to add import

statements in your code. Find the line that begins **import logging**. Create a new line just below it, and copy and paste the following code:

```
import csv
import requests
import io
import calendar
```

- Within the **CaptureZodiacSignIntentHandler**, in the **handle()** function, create a new line and copy and paste the following code:

  #Enter Your URL Here

```
url = "https://docs.google.com/spreadsheets/d/e/2PACX-1vQvr5YePK4pXg0rUOZYEDCh_
response = requests.get(url)
csv_content = response.content
row = csv_content.decode('utf-8').splitlines()
rows = row[1:] # excluding the first row
```

- Now as such you have a data frame in control, you shall be able to **implement a code to extract the zodiac sign** from the data frame relevant to the users date of birth.

- For the provided Zodiac Signs Csv link follow the below steps to extract the data:

  - Create a new function above **handle(self, handler_input)** function named filter Copy and paste the following code:

```
def filter(self, X):
    date = X.split()
    month = date[0]
    month_as_index = list(calendar.month_abbr).index(month[:3].title())
    day = int(date[1])
    return (month_as_index,day)
```

  - Inside **handle()** method copy paste the following code:

```
zodiac = ''
month_as_index = list(calendar.month_abbr).index(month[:3].title())
usr_dob = (month_as_index,int(day))
for sign in rows:
    start, end , zodiac = sign.split(',')
    if self.filter(start) <= usr_dob <= self.filter(end):
        zodiac = zodiac
        break
```

- After the necessary implementation. Within the CaptureZodiacSignIntentHandler, in the handle() function, find the line that begins with **speak_output** . Replace that line with the following code

```
speak_output = 'I see you were born on the {day} of {month} {year}, which means that your zodiac sign will be {zodiac}.'.format(month=month, day=day, year=year, zodiac=zodiac)
```

- Your **CaptureZodiacSignIntentHandler** class should now look like:

```
class CaptureZodiacSignIntentHandler(AbstractRequestHandler):
  """Handler for Hello World Intent."""
. . def can_handle(self, handler_input):
      # type: (HandlerInput) -> bool
……return ask_utils.is_intent_name("CaptureZodiacSignIntent")(handler_input)
  def filter(self, X):
      date = X.split()
      month = date[0]
      month_as_index = list(calendar.month_abbr).index(month[:3].title())
      day = int(date[1])
      return (month_as_index,day)
  def handle(self, handler_input):
      # type: (HandlerInput) -> Response
      slots = handler_input.request_envelope.request.intent.slots
      year = slots["year"].value
      month = slots["month"].value
      day = slots["day"].value
      url =
"https://docs.google.com/spreadsheets/d/e/2PACX-1vQvr5YePK4pXg0rUOZYEDCh_KMa
8gG-E8o7sFjD_Ngww2L2mpXz6Olak7ARSzd9Ng/pub?gid=1494965002&single=true&out
put=csv"
      response = requests.get(url)
      csv_content = response.content
      row = csv_content.decode('utf-8').splitlines()
      rows = row[1:] # excluding the first row
      zodiac = ''
      month_as_index = list(calendar.month_abbr).index(month[:3].title())
      usr_dob = (month_as_index,int(day))
      for sign in rows:
          start, end , zodiac = sign.split(',')
```

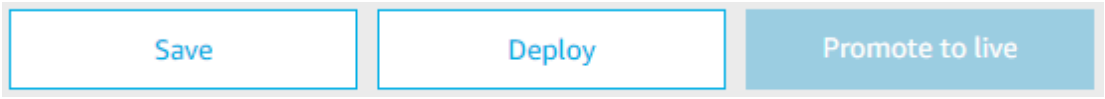
```
        if self.filter(start) <= usr_dob <= self.filter(end):
            zodiac = zodiac
            break
    speak_output = 'I see you were born on the {day} of {month} {year}, which means that your zodiac sign will be {zodiac}.'.format(month=month, day=day, year=year, zodiac=zodiac)
    return (
        handler_input.response_builder
            .speak(speak_output)
            # .ask("add a reprompt if you want to keep the session open for the user to respond")
            .response
    )
```

- Scroll down in the code until you find the line that begins **sb = SkillBuilder()**.
- Replace the line **sb.add_request_handler(HelloWorldIntentHandler())** with **sb.add_request_handler(CaptureZodiacSignIntentHandler())**

Your handler code at the bottom of the file should now look like:

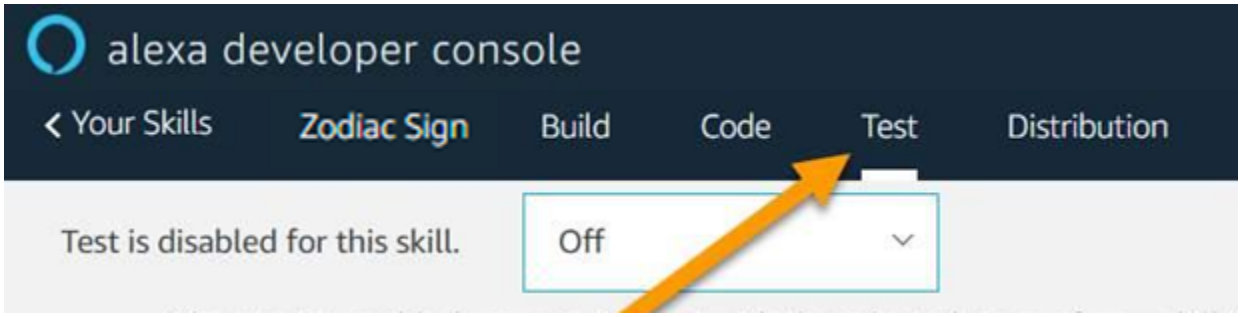
```
sb = SkillBuilder()
sb.add_request_handler(LaunchRequestHandler())
sb.add_request_handler(CaptureZodiacSignIntentHandler())
sb.add_request_handler(HelpIntentHandler())
sb.add_request_handler(CancelOrStopIntentHandler())
sb.add_request_handler(SessionEndedRequestHandler())
sb.add_request_handler(IntentReflectorHandler()) # make sure IntentReflectorHandler is last so it doesn't
override your custom intent handlers
sb.add_exception_handler(CatchAllExceptionHandler())
lambda_handler = sb.lambda_handler()
```

- Click Save and Deploy. Because of the new handler, your skill will take a few moments to deploy.

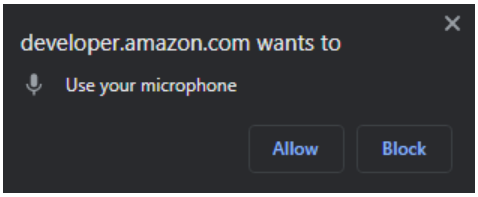


Step 6: Test your skill inside the developer console

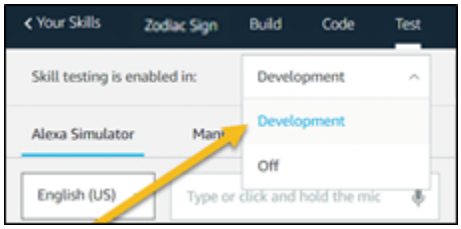
Now it is time to test the skill. Start by activating the test simulator.



An alert may appear requesting to use your computer's microphone. **Click Allow** to enable testing the skill with your voice, just like if you were talking to an Alexa-enabled device.

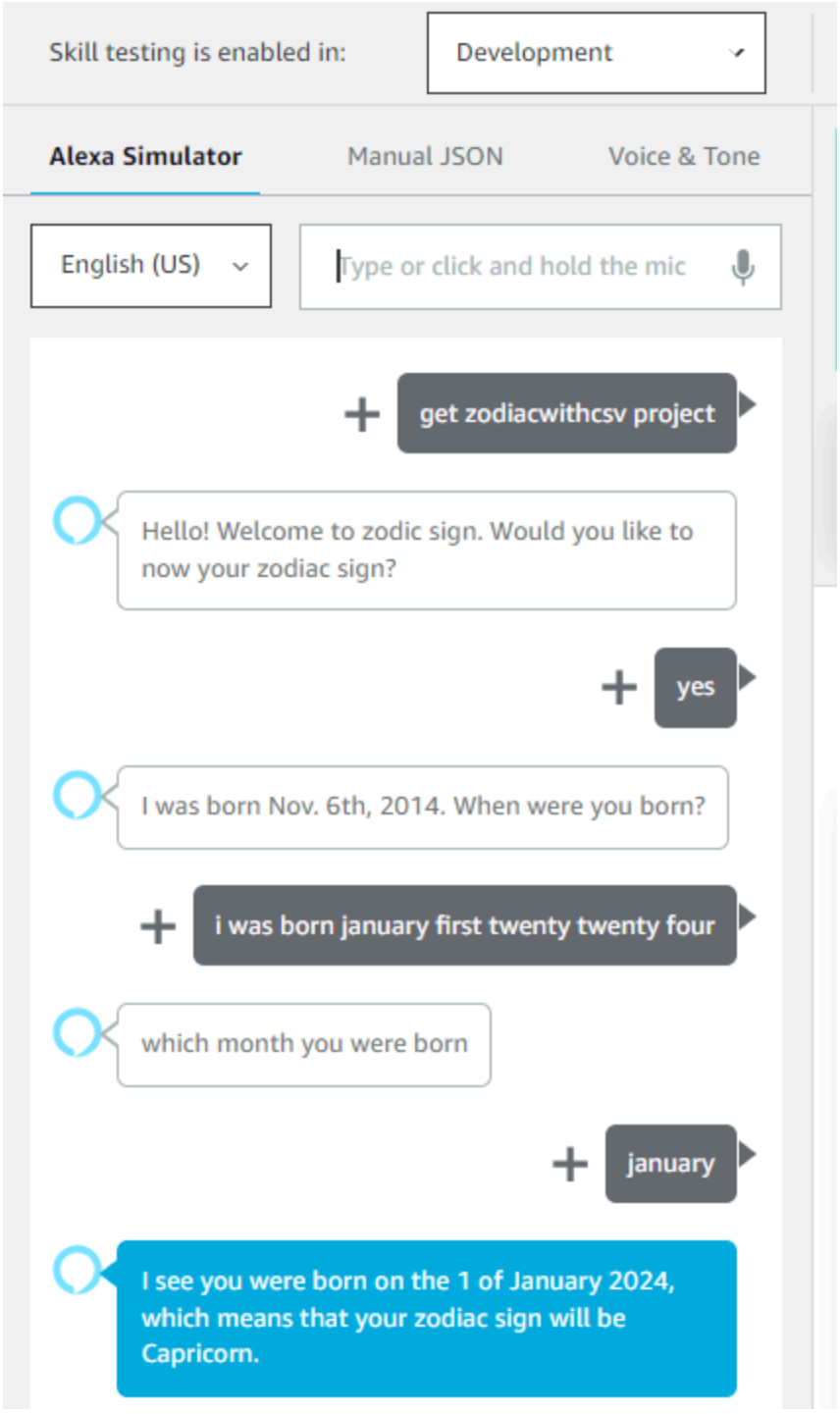


From the drop-down menu at the top left of the page, select **Development.**



There are two ways to test your skill in the console. The user can type the **"open invocation name"** (E.g. open get Zodiac Sign) not case sensitive, into the box at the top left and press ENTER, **or** click and hold the microphone icon and say, "**alexa open get hackathon project**" As shown in the below image, **Be precise—spelling matters!**

Alexa should respond with, "I see you were born on the {day} of {month} {year}, which means that your zodiac sign will be {zodiac}." As shown in the below image.



Go ahead and test what happens if you provide only year, day & year, or other combinations. Alexa should prompt you for any slot values that you omit.

Index

2nd Approach :

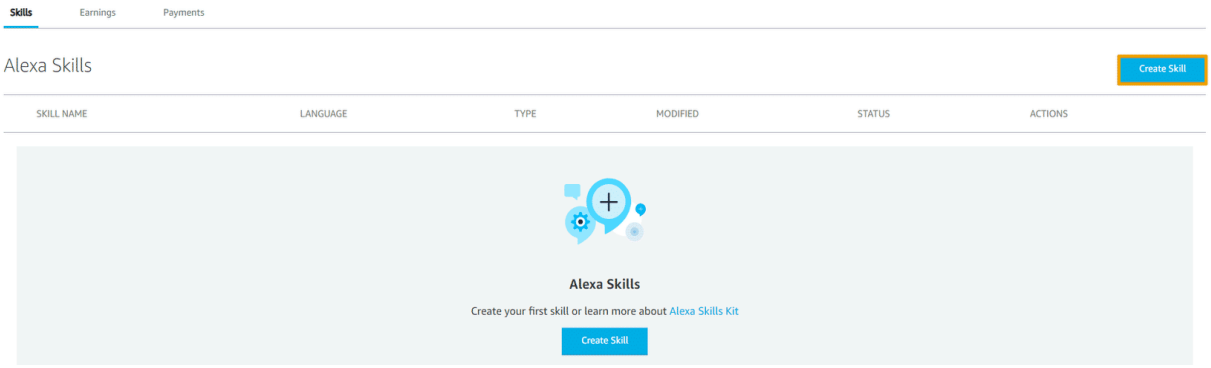
Note:

- The steps outlined above will guide you through the process of creating a zodiac sign intent in Alexa console.
- Repeat steps 3-4-5 to build another intent in the same skill.
- Alexa handles switching between the multiple intents.
- For each intent, a minimum of 50 utterances shall be provided with 3 slots and the respective speech prompts (slot dialogs).

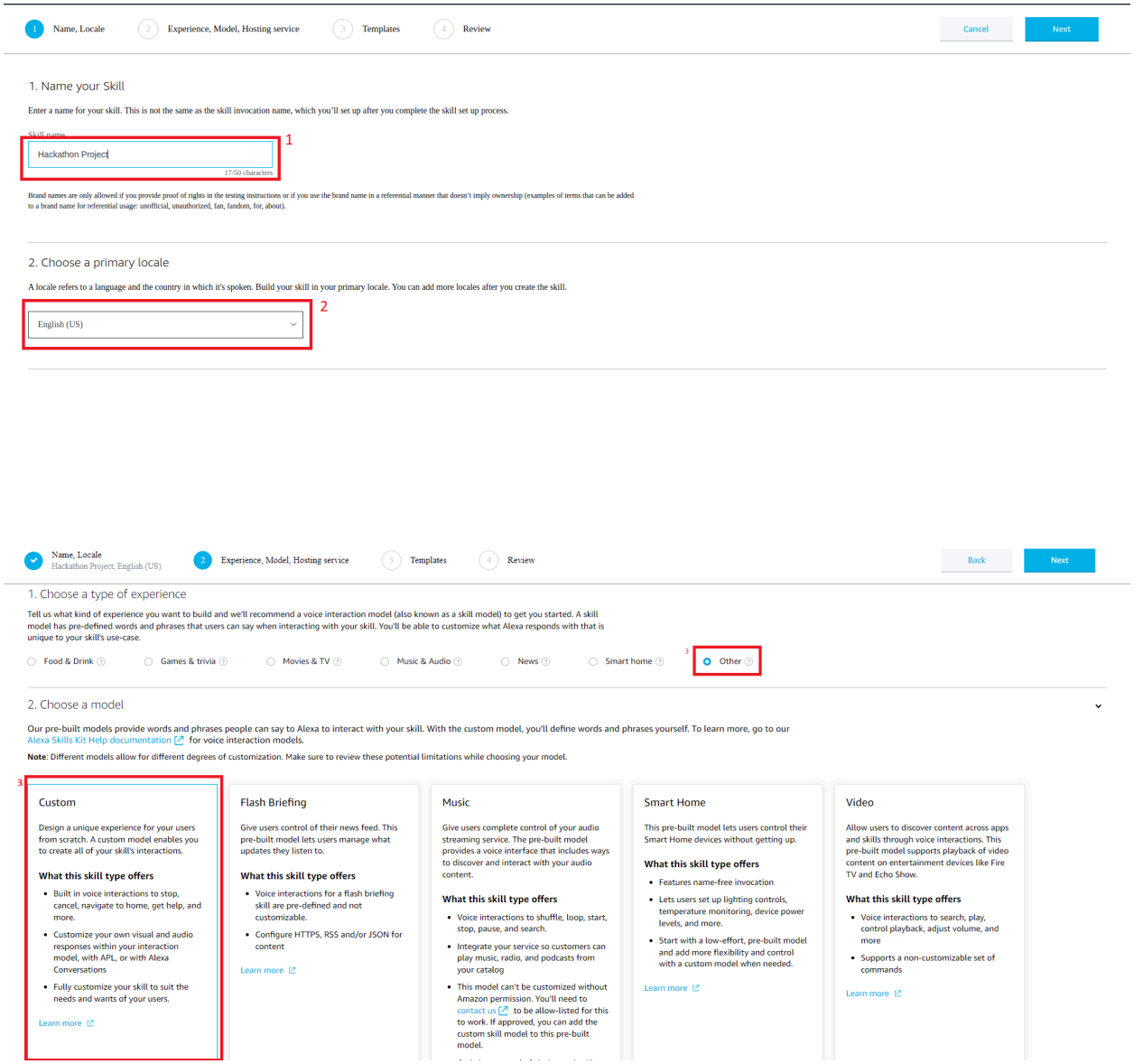
Step1: Login

To get started, log into the Alexa developer console with your Amazon Developer account. If you do not have an account, click here to create one and then login to the developer console.

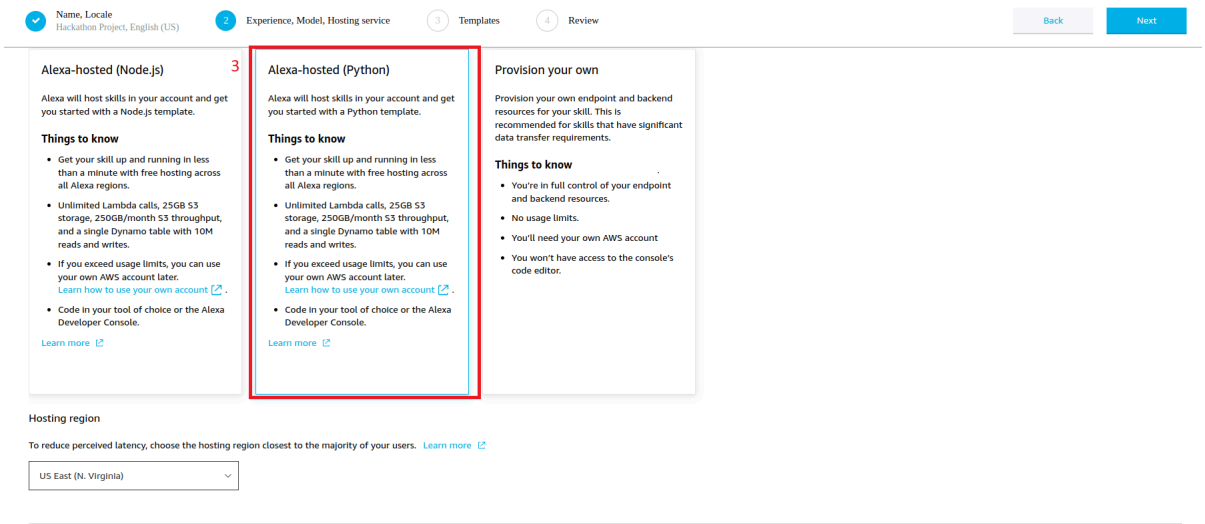
Step2: Create your skill



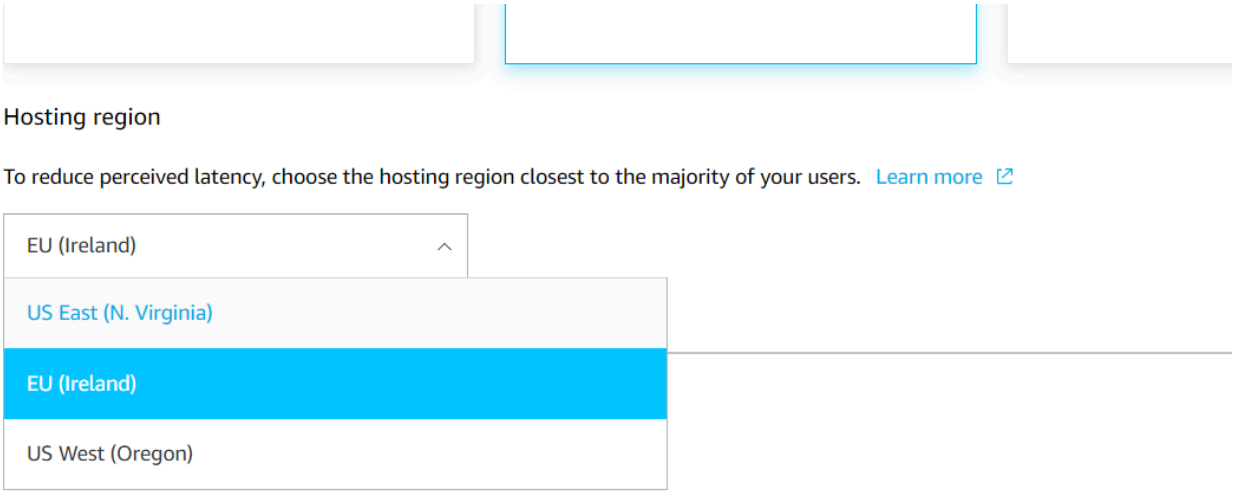
- Click Create Skill on the right-hand side of the console. A new page displays.
- Enter your preferred name of skill (e.g. Hackathon Project) In the **Skill name** field and leave the **Default language** set to **English (US)**.
- You are building a custom skill. **Choose a model to add** to your skill, select **Custom** and **choose a method to host your skill's backend resources**, select **Alexa-Hosted (Python). Refer below image.**



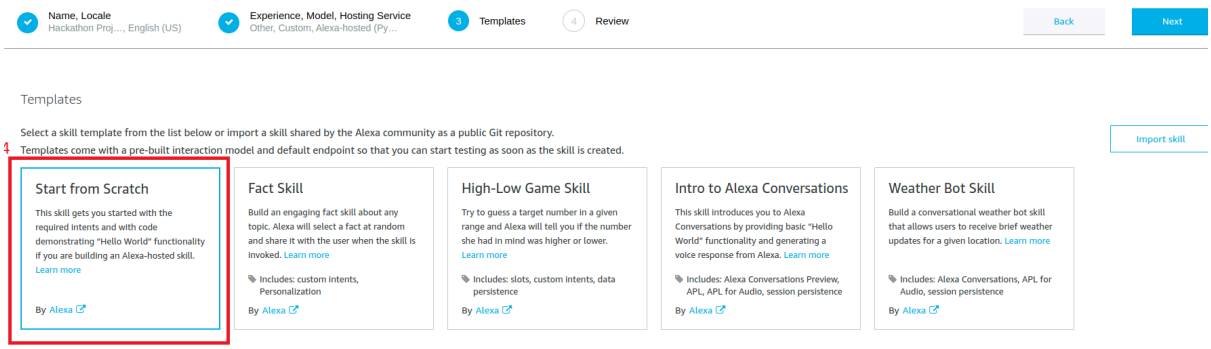
- Choose a type of experience as other and choose a model as custom since we are building our custom model and scroll down.



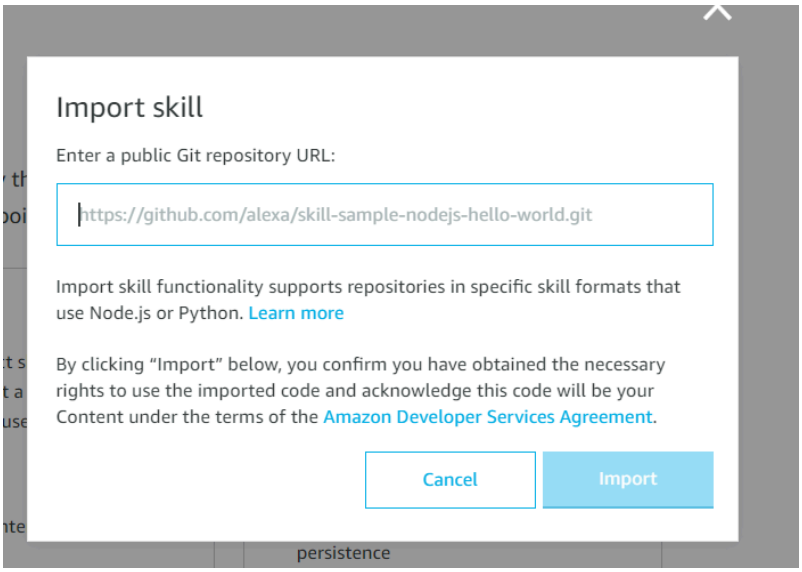
- Scroll down and select Alexa-hosted (python) and hosting region EU (ireland) as below.

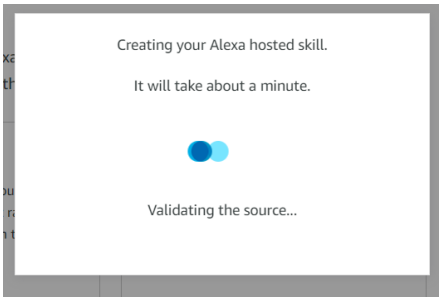


- Choose a template to select a Hello World Skill From the given Templates and at the top of the page, Click Choose as per the below image.

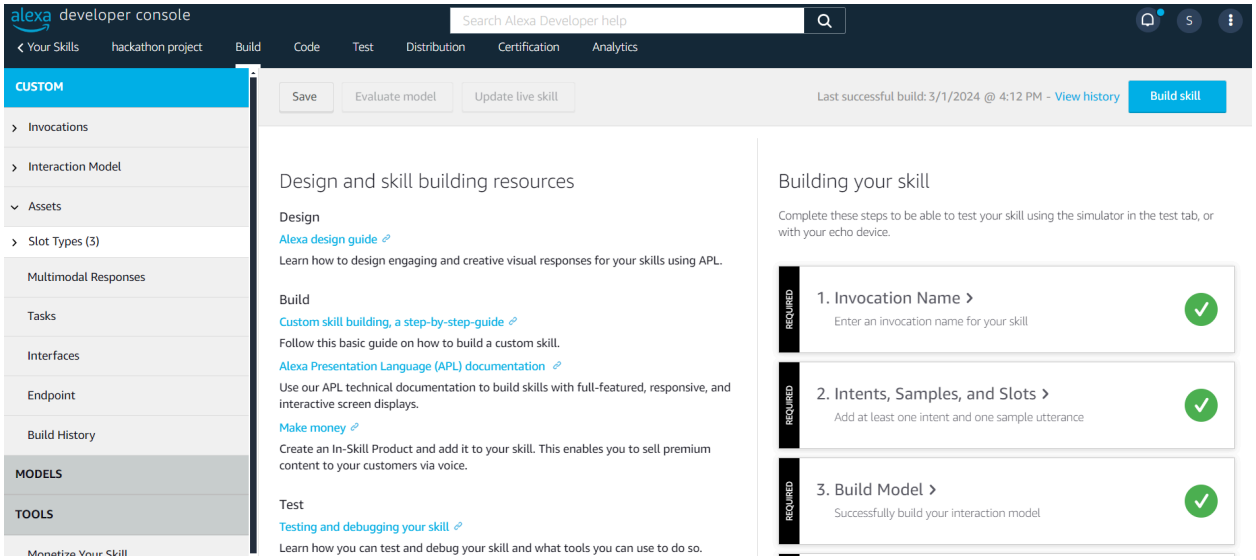


- Choose a template Start from Scratch and Top Right side click on import skill and there will be a pop up. paste the git repo url " https://github.com/sumanthkalyan21/alexachatbot.git " in to the space provided and import.

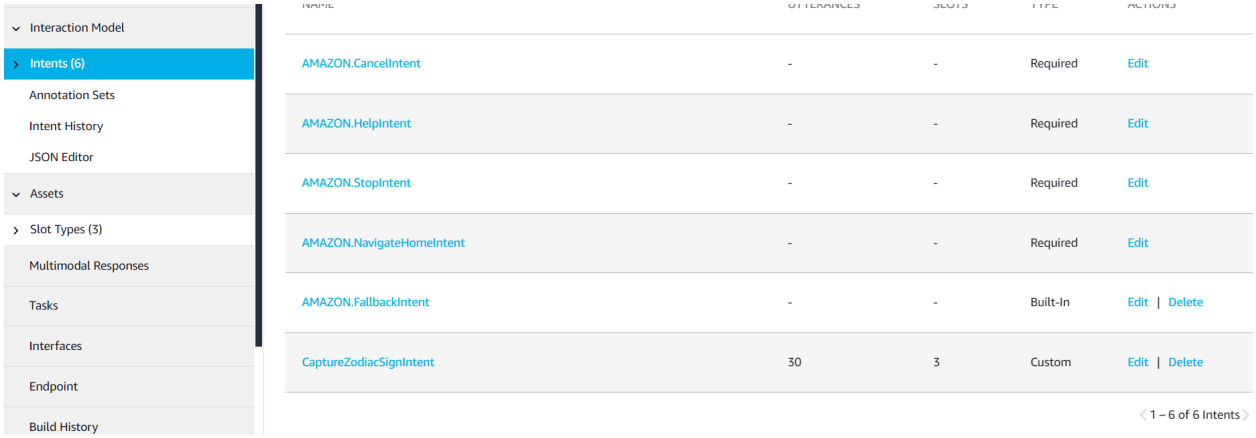




- It takes a few moments for AWS to provision resources for your skill. When this process completes, move to the next section.



- Click on the Invocation tab and change the **1.Invocation name** as per your skill as shown in the below image. As this invocation name will be used in testing.
  Note: Invocation is the act of beginning an interaction with a custom skill, for example the phrase: "Alexa, get hackathon project" tells Alexa to use the hackathon project skill. The keyword "get hackathon project" is known as invocation name *(you can give any name for the skill or invocation)*.



Step 3: Create intent and slots to capture information

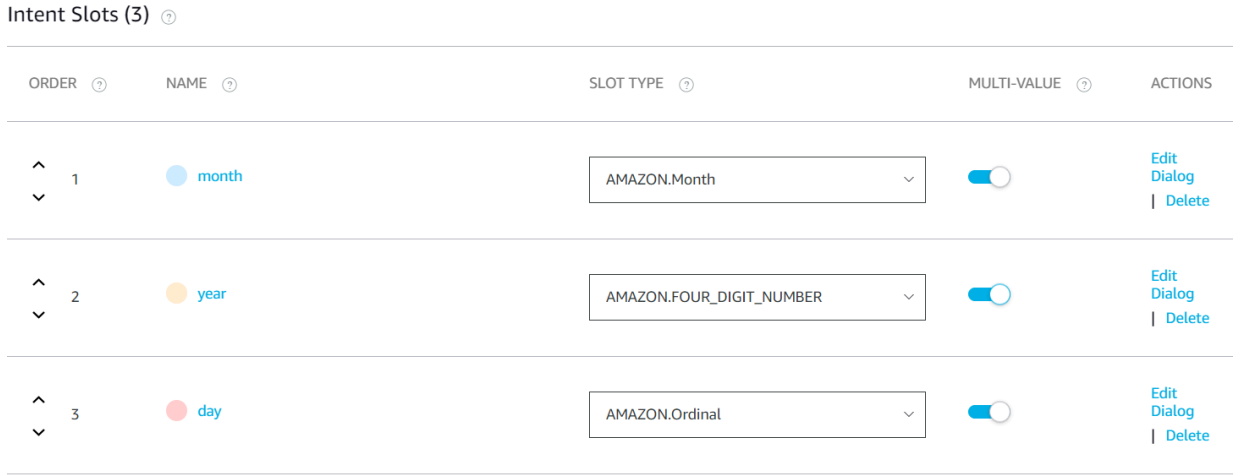
In this section, you will make the skill more useful by having it ask the user for their date of birth. So that when the user responds, the skill will understand and repeat the user's zodiac sign back to them.

To do this, you will need to use **utterances, intents, and slots**. You will also learn how to use **dialog management** to have your skill automatically ask follow-up questions to collect the

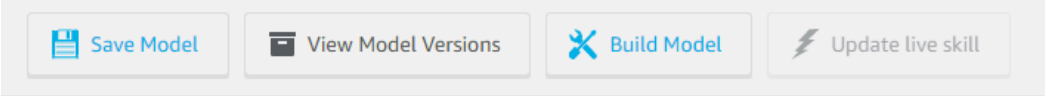
required information. **For example**, if the user says, "**I was born July 12th**," dialog management will automatically ask the user **what year they were born**.

At the end of this module, your **Zodiac Sign** intent will be able to ask the user a question and listen for the answer further to respond to the user.

Note: Since we had imported our skill the Intent, Slots and Utterances will also be imported, here you need to verify them and create as many utterances as requested.



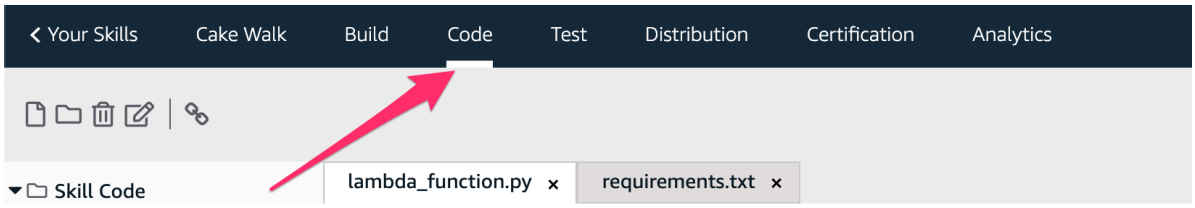
- At the top of the page, click **Save Model** and **Build Model.**



- When you click **Build Model**, your skill starts to build the training data that will help Alexa know how to map what the user says to your skill's intents. It may take a minute for the model to build.

Step 4: Let's Start Building the Model in the Backend.

In this step, you will update your backend code to respond to the user when they open the skill.

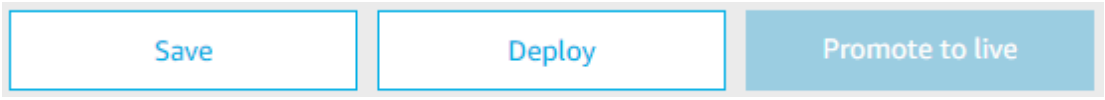


Click the **Code tab**, you can see an online code editor with some files already set up for you to get started. In particular, you'll see the following three files in the lambda sub-directory:

- **lambda_function.py:** This is the main entry point of the backend service. All the request data from the Alexa intent is received here and is supposed to be returned from this file only.
- **requirements.txt:** This file contains the list of Python packages that are being used in this project.
- **utils.py:** This file contains some utility functions required for the lambda function.

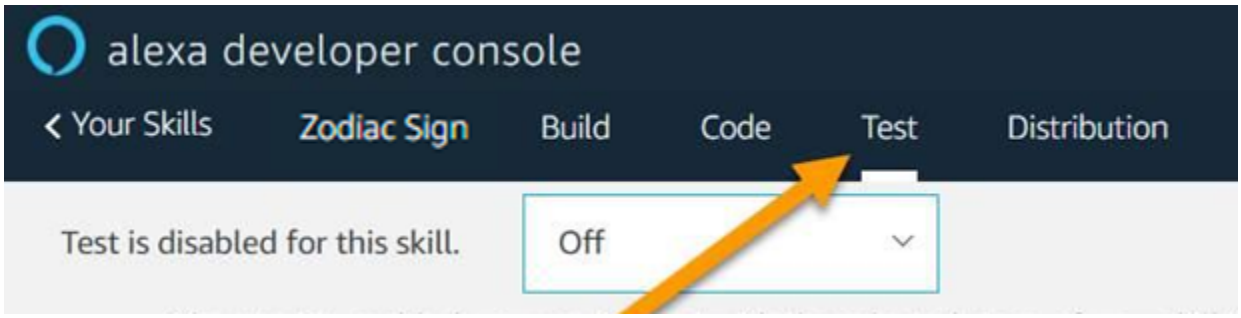
Note: Here this lambda_function.py and requirements.txt and utils.py has been imported, here verify the code according to the 1st approach and go for deployment.

- Click Save and Deploy. Because of the new handler, your skill will take a few moments to deploy.

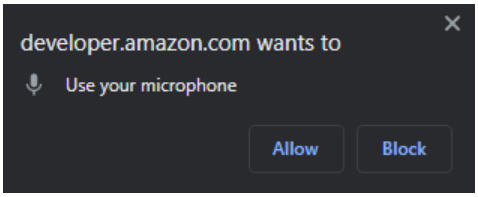


Step 5: Test your skill inside the developer console

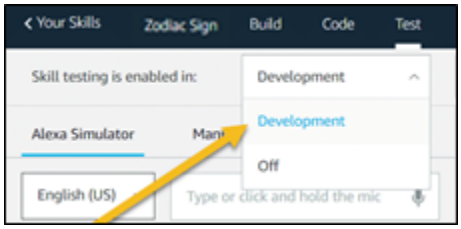
Now it is time to test the skill. Start by activating the test simulator.



An alert may appear requesting to use your computer's microphone. **Click Allow** to enable testing the skill with your voice, just like if you were talking to an Alexa-enabled device.

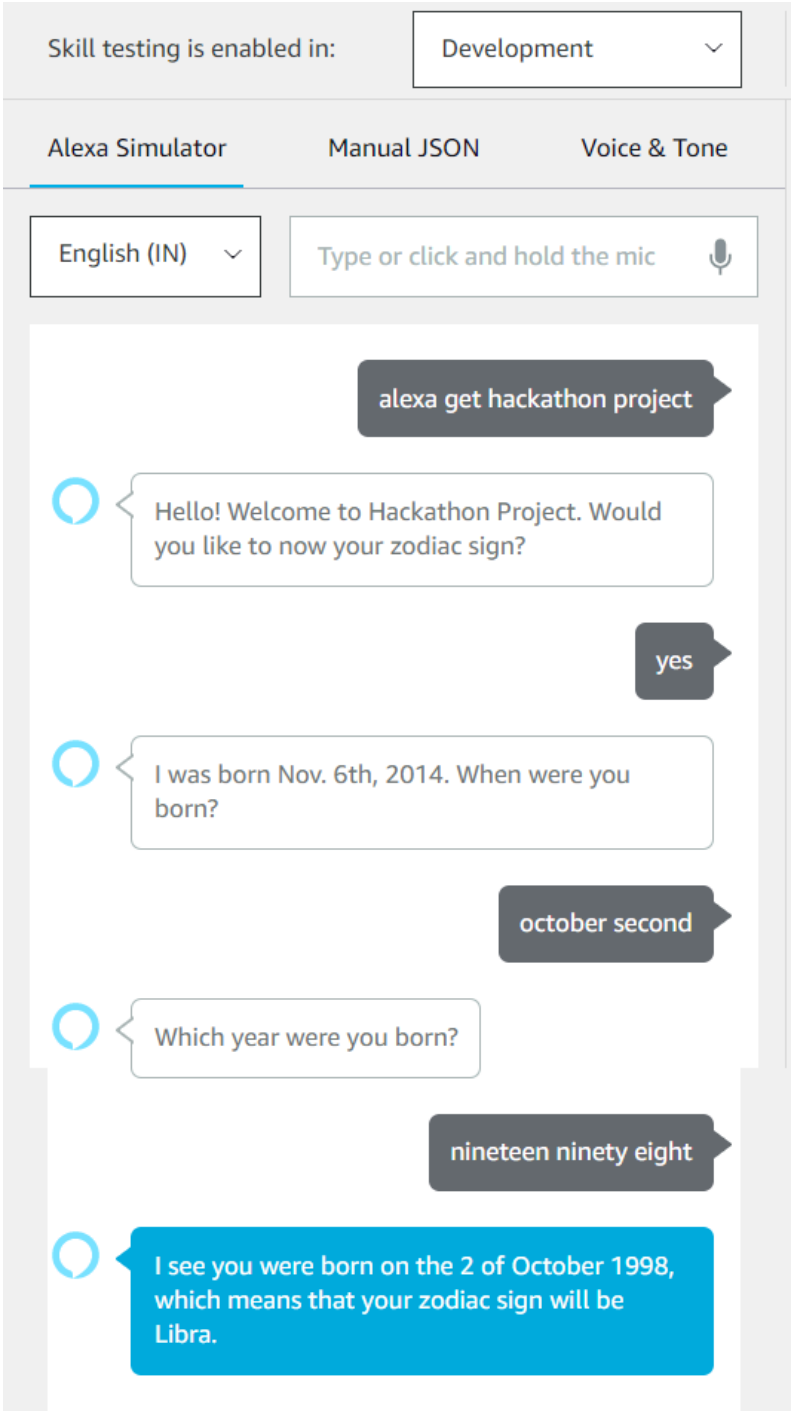


From the drop-down menu at the top left of the page, select **Development.**



There are two ways to test your skill in the console. The user can type the **"open invocation name"** (E.g. open get Zodiac Sign) not case sensitive, into the box at the top left and press ENTER, **or** click and hold the microphone icon and say, "**alexa open get hackathon project**" As shown in the below image, **Be precise—spelling matters!**

Alexa should respond with, "I see you were born on the {day} of {month} {year}, which means that your zodiac sign will be {zodiac}." As shown in the below image.



Go ahead and test what happens if you provide only year, day & year, or other combinations. Alexa should prompt you for any slot values that you omit.